

MAGAZINE **BSD**

FOR NOVICE AND ADVANCED USERS

ADDING ZFS TO THE FreeBSD

DUAL-CONTROLLER STORAGE CONCEPT

REUSING THE OpenBSD ARC4RANDOM

IN MULTI-THREADED USER
SPACE PROGRAMS

FREEBSD FLAVOURS GHOSTBSD

SECURE VPNs WITH GRE / STRONGSWAN

SMALL BUSINESS NETWORKS WITH FreeBSD

HOW TO INSTALL XFCE 4.12 DESKTOP ON NETBSD 7

INTERVIEW WITH

FERNANDO RODRÍGUEZ,
CO-FOUNDER OF KEEP CODING

VOL. 10 NO. 04

ISSUE 04/2016 (80)

1898-9144

SOMETHING XL IS COMING IN APRIL



ENTERPRISE-CLASS HARDWARE, RUNNING
THE WORLD'S MOST POPULAR OPEN SOURCE
STORAGE OPERATING SYSTEM.

For more information on the FreeNAS Mini,
visit **ixsystems.com/mini** today.

Then, check back in April for something XL...

Dear Readers,

Finally, we have a beautiful month of May. It's warm, the sun is shining, plants and trees are blooming and we can enjoy all of that thanks to many public holidays. I hope that in your countries this month brings many days off as well as it does in Poland and Denmark. If some of them are still ahead of you, I hope this issue will make your time off more pleasant.

We will start with a very long "Adding ZFS to the FreeBSD Dual-Controller Storage Concept" article by Mikhail E. Zakharov. Grab a cup of coffee!

The second article, "Secure VPNs with Gre and Strongswan for Small Business Networks with FreeBSD" by Antonio Francesco Gentile, will explain how, with secure, reliable, and confidential communication, it is possible to reduce costs and improve the efficiency of production processes..

Now it is time for OpenBSD. Here you will read about "Reusing the OpenBSD arc4random in Multithreaded User Space Programs" by Sudhi Herle. Upgrade your OpenBSD to the latest version and start your testing.

Would you like to know "HOW to Install the XFCE 4.12 Desktop on NetBSD 7"? Curt McIntosh will show you how!

A lot of BSDs in this issue, huh? Well, here is the last one. It's George Bungarzescus' debut article about GhostBSD. Enjoy "FreeBSD Flavors. Do We Need Them? Today...GhostBSD. A (not too deep) journey to GhostBSD - desktop and enterprise options - compared to pure FreeBSD".

In our interview, Fernando Rodríguez, Co-founder of KeepCoding tells us in the interview about his school, learning to program and that it pays off to be hard working.

And in the end, as always, Rob Somerville and his comment on copyright infringement.

Enjoy the whole issue and beautiful long days in May!

Marta & BSD Team

MAGAZINE BSD

Editor in Chief:

Marta Ziemianowicz

marta.ziemianowicz@software.com.pl

Contributing:

Mikhail E. Zakharov, Antonio Francesco Gentile, Fernando Rodríguez, George Bungarzescu, Sudhi Herle, Curt McIntosh and Rob Somerville.

Top Betatesters & Proofreaders:

Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth, Ben Milman, Mark VonFange and David Carlier

Special Thanks:

Annie Zhang

Denise Ebery

DTP:

Marta Ziemianowicz

Senior Consultant/Publisher:

Paweł Marciniak

pawel@software.com.pl

CEO:

Joanna Kretowicz

joanna.kretowicz@software.com.pl

Publisher:

Hakin9 Media SK 02-676 Warsaw, Poland Postepu 17D Poland worldwide
publishing editors@bsdmag.org www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail:
editors@bsdmag.org.

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.

CONTENTS

News

BSD World Monthly News **4**

by Marta Ziemianowicz

This column presents the latest news coverage of breaking news events, product releases and trending topics.

ZFS

Adding ZFS to the FreeBSD Dual-Controller Storage Concept **14**

by Mikhail E. Zakharov

We should name our small project. Something in honor of Beastie the daemon and the large BSD operating system family should be reasonable, so let's call our storage project the BeaST or even shorter, the BST.

The environment for our testing purposes will be similar to previous ones. It is still my laptop which runs Oracle VM VirtualBox and a USB memory-stick to store shareable virtual drives and slow down their IO.

The FreeBSD Corner

Secure VPNs with Gre and Strongswan for Small Business Networks with FreeBSD **41**

by Antonio Francesco Gentile

Communication is a competitive advantage for any company. With secure, reliable, and confidential communication it is possible to reduce costs, and improve the efficiency of production processes. In this article we will see how to implement them in FreeBSD.

OpenBSD

Reusing the OpenBSD arc4random in Multi-threaded User Space Programs **66**

by Sudhi Herle

Recently, a friend asked me for some help in speeding up crypto operations in her multi-threaded (pthreads) Linux program. She was using the standard /dev/urandom interface to generate random

bytes. And, profiling showed that reading from /dev/urandom took a lot of time. I recalled that OpenBSD folks have a high quality userspace cryptographic random number generator called arc4random. But to the best of my knowledge, it was not generally available for use in any Linux program. And so, it seemed like an excellent weekend project to create a portable version of the OpenBSD arc4random(3) generator for use in multi-threaded programs.

NetBSD

HOW TO install the XFCE 4.12 Desktop on NetBSD 7 **77**

by Curt McIntosh

For a lightweight functional desktop on NetBSD, install XFCE. As root, perform the following steps. This covers 32 and 64 bit x86 hardware. Since NetBSD essentially runs on everything, simply adjust the repository path to your architecture from the list...

GhostBSD

FreeBSD Flavors. Do We Need Them? Today...GhostBSD **84**

by George Bungarzesu

A (not too deep) journey to GhostBSD - desktop and enterprise options - compared to pure FreeBSD

Interview

Interview with Fernando Rodríguez, Co-founder of KeepCoding **90**

by Marta Ziemianowicz, Marta Strzelec & Marta Si-enicka

Rob's Column **97**

by Rob Somerville

Baroness Neville-Rolfe DBE CMG, in the recently released UK government document "Criminal Sanctions for Online Copyright Infringement" mandates a 10 year prison sentence for serious instances of copyright infringement. This intends to bring the penalties in line with those found guilty of copyright breaches in respect to physical goods. Will this amendment help to reduce piracy?

BSD Certification

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

? WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BDSP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BDSP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

✓ WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BDSP exams are yet to be determined.

Payments are made through our registration website:
<https://register.bsdcertification.org/register/payment>

i WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:
<https://register.bsdcertification.org/register/get-a-bsdcg-id>

Linux greybeards release beta of systemd-free Debian fork



Devuan 'Jessie' betas debut in the name of 'Init freedom'

The effort to create a systemd-free Debian fork has borne fruit, with a beta of “Devuan Jessie” appearing in the wild.

Devuan came into being after a rebellion by a self-described “Veteran Unix Admin collective” argued that Debian had betrayed its roots and was becoming too desktop-oriented. The item to which they objected most vigorously was the inclusion of the systemd process manager. The rebels therefore decided to fork Debian and “preserve Init freedom”. The group renamed itself and its distribution “Devuan” and got to work, promising a fork that looked, felt, and quacked like Debian in all regards other than imposing systemd as the default Init option.

The group initially promised to deliver in Spring 2015. Alphas circulated during 2015, and in recent days betas have appeared here. Versions for the Raspberry Pi, Banana Pi and AMD64 are on offer.

Kudos, though, to the group for getting it out there! Now to see if there is really a groundswell of support for the cause of “Init freedom”, as the greybeards name their cause.

http://www.theregister.co.uk/2016/04/29/systemd_free_debian_fork_devuan_reaches_beta/

Node.js Version 6 LTS Released — World's Fastest Growing Open Source Platform



To ensure an improvement performance experience for its 3.5 million users, Node.js Foundation has released Node.js Version 6 with Long Term Support. This release supports 93 percent of the ECMAScript 6 standard and uses Google's V8 version 5.0 for the JS engine.

Node.js, the JavaScript runtime, has hit version 6. The new Node.js v6 brings along lots of performance and security enhancements.

Node.js v6 is released as a long term support (LTS) release with an aim to provide a high-quality solution to the developers without compromising on the consistency front. Thus, Node.js 6.x will continue receiving official support until April 2018, and then grab only maintenance updates until April 2019.

This Node.js 6 LTS release has also marked the end of Long Term Support for the Node.js 0.12 branch. As a result, currently, the officially supported versions are 4.x and 6.x.

Note that the Long Term Support for Node.js 4.x will end in April 2017. So, you have plenty of time to make the move to version 6.x LTS.

However, Node.js 6.x will continue to remain the recommended Node.js version for the production usage because even though Node.js 6.x is a stable release, it will also contain new JS features that are under testing.

Node.js also supports 93% of the ECMAScript 6 standard that was released in June 2015, which is much higher as compared to the 56% support provided by Node.js 5.x.

The other major changes in Node.js 6.x include the usage of Google's V8 version 5.0 for the JS engine and other improvements.

<http://fossbytes.com/node-js-version-6-lts-released-features/>

FreeBSD Mastery: Advanced ZFS by Michael W Lucas & Allan Jude



Michael W Lucas, critically acclaimed author of many BSD books, introduces the long awaited FreeBSD Michael W Lucas, critically acclaimed author of many BSD books, introduces the long awaited FreeBSD Mastery: Advanced ZFS co-written by Allan Jude. This book is the follow-up to FreeBSD Mastery: ZFS, which released last year.

ZFS improves everything about systems administration. Once you peek under the hood, though, ZFS' bewildering array of knobs and tunables can overwhelm anyone. ZFS experts can make their servers zing - and now you can too, with FreeBSD Mastery: Advanced ZFS.

This small book teaches you to:

- Use boot environments to make the riskiest sysadmin tasks boring
- Delegate filesystem privileges to users
- Containerize ZFS datasets with jails
- Quickly and efficiently replicate data between machines
- Split layers off of mirrors
- Optimize ZFS block storage
- Handle large storage arrays
- Select caching strategies to improve performance
- Manage next-generation storage hardware
- Identify and remove bottlenecks
- Build screaming fast database storage
- Dive deep into pools, metaslabs, and more!

Whether you manage a single small server or international datacenters, simplify your storage with FreeBSD Mastery: Advanced ZFS.

<https://www.freebsdnews.com/2016/04/22/freebsd-mastery-advanced-zfs/>

Chromium OS for Raspberry Pi 3 officially released and here's everything you should know



Supercharge your Raspberry Pi 3 right away

A separate Chromium OS update has been made for Raspberry Pi 3 owners and for those who wanted to run the operating system on the latest iteration of Raspberry Pi 3; however, they need to know the following things before they can go ahead and download the Chromium OS for Raspberry Pi 3 0.5 binary image.

Here is what is going to be included in Chromium OS for Raspberry Pi 3.

Chromium OS for Raspberry Pi 3 includes several improvements and features. One of them happens to be the Linux 4.2.8-ckt8 kernel with a reduced size, as well as BFS tweaks for improved latency, and less debugging output. Moreover, it also brings multiple improvements to the sound driver, as well as better storage performance; this is due to the new BFQ hierarchical scheduler and on demand governor tweaks.

However, performance is not the only thing that is going to be rated here, since a fluid and clutter free user experience is also required in order to ensure that this update is recommended to other Raspberry Pi 3 owners. As of the coming of this experience, it has improved significantly, and offers fixes for the Kiosk Mode. Additionally, it also brings fixes to the VC4 GPU driver and various video modes, especially for those running Chromium OS for Raspberry Pi 3 on non-1080p displays.

Raspberry Pi 3 Model B was released on February 29, 2016, and featured a built-in Wi-Fi and Bluetooth chip. Not only this, but the logic board also sported a 64-bit quad-core ARM Cortex-A53 processing core, which would eventually deliver significant performance as compared to the logic board's predecessors. While there are a ton of improvements introduced, there are also more than a few limitations that you should be informed about.

First of all, the on-board Wi-Fi of the Raspberry Pi 3 Model B computer is not yet supported, and Netflix videos cannot be streamed as well. Another drawback is that HTML5 playback works only on websites that also offer Flash as an alternative to HTML5 video. If you are absolutely comfortable with these limitations, then you can proceed to download the binary image right now.

<http://www.techworm.net/2016/04/chromium-os-raspberry-pi-3-officially-released-heres-everything-know.html>

Blog: Larry the BSD Guy's last column [FossForce]



Larry the BSD Guy, also known as the Free Software Guy, is hanging up his jersey to pursue another of his own writing interests. Read his final column from Fossforce, this one featuring LinuxFest NorthWest, an annual open-source trade show held in Bellingham, Washington. Thank you for your contributions to the free and open source community, and we wish you the best in future endeavors.

It is a sad day at the FOSS Force office. Larry Cafiero says goodbye and walks off into the sunset.

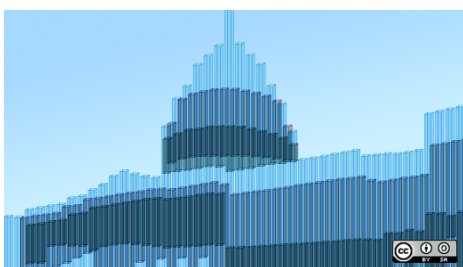
This weekend, the Grand Old Man (or Woman — take your pick) of Linux expos in North America takes place in the upper left corner of the United States.

For over a decade and a half, LinuxFest Northwest has flown the flag literally in Microsoft's backyard, an annual open source event held the last weekend in April in Bellingham, Wash. LFNW features presentations and exhibits on various free and open source topics, as well as Linux distributions and applications. It usually has something for everyone from the novice to the professional.

It has a special place in my heart as well. While I think that SCALE is the best show on the continent for obvious reasons (the SCALE Publicity Team is solely responsible, he says in jest), LFNW is my favorite show to attend, not only because of the history but because of the community vibe the show gives off at an expo that has refused to give in to the creeping corporatism to which other shows have succumbed

<https://www.freebsdnews.com/2016/04/22/larry-the-bsd-guys-last-column-fossforce/>

A fresh look at the U.S. draft policy on 'federal sourcing'



The policy both reaffirms and broadens a goal laid out in the Administration's Second Open Government National Action Plan for "improved access to custom software code developed for the Federal Government." The Plan emphasized use of (and contributing back to) open source software to fuel innovation, lower costs, and benefit the public. It also furthers a long-standing 'default to open' objective going back to the early days of the Administration.

The draft policy features several components. First, it provides a recommended "3-step process" on software procurement considerations to minimize procurement of custom-developed software. Second, it establishes requirements for releasing code in the public domain or as open source software, replicating early efforts by agencies to have in place policies to release code developed in-house.

Finally, covered agencies must deliver for reuse custom code produced in the performance of a federal government agreement and, subject to certain exceptions, make it broadly available government-wide. As widely reported, each covered agency will participate in a pilot program to "release at least 20 percent of its newly-developed custom code each year as open source," using existing Open Source Initiative licenses, but leaves room for additional licenses as necessary.

<https://opensource.com/government/16/4/draft-policy-federal-sourcing>

GhostBSD 10.3 Enoch ALPHA1 now available



The developers of GhostBSD have made available the first 10.3 ALPHA. This GhostBSD distribution is code-named "Enoch". Notable changes are added support for ZFS and UDF, as well as the base system being FreeBSD 10.3.

Yes we skipped 10.2 for 10.3 since FreeBSD 10.3 was coming, we thought we should wait for 10.3. This is the first ALPHA development release for testing and debugging for GhostBSD 10.3, only as MATE been released yet which is available on SourceForge and for the amd64 and i386 architectures.

What's new

- GhostBSD now support ZFS and UFS.
- The installer support encryption for ZFS
- GhostBSD Software will be updated Quarterly which will bring more stability to GhostBSD still user will be able to change it to latest to have the latest software up date.

What changed

- The installer did have a big refacing plus a new slide
- There is been some important fix to Networkmgr
- There is been some speed improvement with Networkmgr
- Some UI improvement and speed improvement with Update Station
- Mate 1.12
- New Grub theme

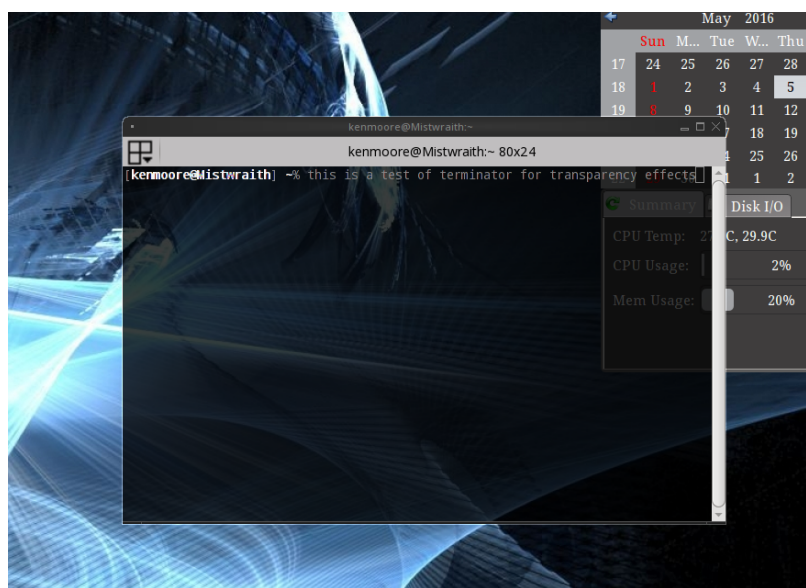
What has been fixed

- Language wish GhostBSD is installed should now be fix
- Wifi Problem as been fix with the update of Networkmgr

Official announcement: http://www.ghostbsd.org/10.3_alpha1

<https://www.freebsdnews.com/2016/05/06/ghostbsd-10-3-to-add-zfs-and-udf-support-will-be-based-on-freebsd-10-3/>

PC-BSD's Lumina Desktop 0.9.0 Environment now available



Ken Moore, creator of the Lumina Desktop Environment, has made available version 0.9.0. This version adds a new compositing effect, and more improvements are lined up for version 1.0. Lumina is also part of the PC-BSD project.

“First, I would like to thank everyone for their patience as we continue working toward the first non-beta release of the Lumina desktop. We are still planning on version 1.0.0 getting released later this year (aligning with the FreeBSD 11.0 Release Schedule or earlier), but some issues have come to light that required we adjust

our feature list for version 1.0.0 a bit.”

Official announcement:

<http://lumina-desktop.org/new-release-schedule-and-lumina-desktop-0-9-0-released/>

<https://www.freebsdnews.com/2016/05/06/pc-bsds-lumina-desktop-0-9-0-environment-launches-with-compositing-effects/>

pfSense 2.3-RELEASE Now Available!

The most significant changes in this release is a rewrite of the webGUI utilizing Bootstrap, and the underlying system, including the base system and kernel, being converted entirely to FreeBSD pkg. The pkg conversion enables us to update pieces of the system individually going forward, rather than the monolithic updates of the past. The webGUI rewrite brings a new responsive look and feel to pfSense requiring a minimum of resizing or scrolling on a wide range of devices from desktops to mobile phones.

For the highlights, check out the Features and Highlights video. Past blog posts have covered some of the changes, such as the performance improvements from tryforward, and the webGUI update.

To get to a release, we have closed 760 total tickets. While the majority of these were related to the Bootstrap conversion, 137 are fixed bugs impacting 2.2.6 and earlier releases.

Upgrade Considerations

As always, you can upgrade from any prior version directly to 2.3. The Upgrade Guide covers everything you will need to know for upgrading in general. There are a few areas where additional caution should be exercised with this upgrade.

Known Regressions

- OpenVPN topology change – configuration upgrade code was intended to set upgraded OpenVPN servers to topology net30, rather than the new default of topology subnet. This is not working as intended in some cases, but has been fixed for 2.3.1. In the meantime, editing your OpenVPN server instance and setting the topology to “net30”, will accomplish the same thing and fix it.
- IP aliases with CARP IP parent lose their parent interface association post-upgrade. Go to Firewall>Virtual IPs, edit the affected IP alias, pick the appropriate CARP IP parent, then save and apply changes. Make sure every virtual IP has something shown in the Interface column on firewall_virtual_ip.php.
- Psec IPComp does not work. This is disabled by default. Disable IPComp under VPN>IPsec, Advanced to work around if you have enabled IPComp. Bug 6167
- IGMP Proxy does not work with VLAN interfaces. Bug 6099. This is a little-used component. If you are not sure what it is, you are not using it.

Clear Browser Cache

Due to the many changes in the web interface, clearing your browser cache or doing a forced re-load (shift+refresh) is a good idea after upgrading. If you see any cosmetic problems in the web interface post-upgrade, a stale browser cache is the likely reason.

Packages

The list of available packages in pfSense 2.3 has been significantly trimmed. We have removed packages that have been deprecated upstream, no longer have an active maintainer, or were never stable. A few have yet to be converted for Bootstrap and may return if converted. pfSense software is Open Source

For those who wish to review the source code in full detail, the changes are all publicly available in three repositories on Github. 2.3-RELEASE and is built from the RELENG_2_3_0 branch of each repository.

Supporting the Project

Our efforts are made possible by the support of our customers and the community. You can support our efforts via one or more of the following ways:

- pfSense Store – official hardware, apparel, and pre-loaded USB sticks direct from the source. Our pre-installed appliances are the fast, easy way to get up and running with a fully-optimized system. All are now shipping with 2.3 release installed.
- Gold subscription – Immediate access to past hang out recordings , as well as the latest version of the book after logging in to the members area.
- Commercial Support – Purchasing support from us provides you with direct access to the pfSense team.
- Professional Services – For more involved and complex projects outside the scope of support, our most senior engineers are available under professional services.

<https://blog.pfsense.org/?p=2008>

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES

1.925.240.6652

\$39.95

FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

\$29.95

PC-BSD 9.1 DVD

\$49.95

The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

\$99.95

The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD



Stylish Dress Attire
Look Your Professional Best



Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD \$39.95

CD Set Contains:

- Disc 1** Installation Boot LiveCD (i386)
- Disc 2** Essential Packages Xorg (i386)
- Disc 3** Essential Packages, GNOME2 (i386)
- Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD \$39.95

FreeBSD 9.0 DVD \$39.95

FreeBSD Subscriptions

Save time and \$\$\$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 \$29.95

FreeBSD Subscription, start with DVD 9.1 \$29.95

FreeBSD Subscription, start with CD 9.0 \$29.95

FreeBSD Subscription, start with DVD 9.0 \$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD \$29.95

PC-BSD Subscription \$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) \$39.95

The FreeBSD Handbook, Volume 2 (Admin Guide) \$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) \$59.95

The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 \$79.95

PC-BSD 9.0 Users Handbook \$24.95

BSD Magazine \$11.99

The FreeBSD Toolkit DVD \$39.95

FreeBSD Mousepad \$10.00

FreeBSD & PCBSD Caps \$20.00

BSD Daemon Horns \$2.00



Bundle Specials!
Save \$\$\$

Just Plain Fun
Mousepads & Novelty Horns



BSD Magazine
Available Monthly



For even MORE items
visit our website today!

www.FreeBSDMall.com

Adding ZFS to the FreeBSD Dual-controller Storage Concept

by Mikhail E. Zakharov

We should name our small project. Something in honor of Beastie the daemon and the large BSD operating system family should be reasonable, so let's call our storage project the BeaST or even shorter, the BST.

The environment for our testing purposes will be similar to previous ones. It is still my laptop which runs Oracle VM VirtualBox and a USB memory-stick to store shareable virtual drives and slow down their IO.

We will need to create three virtual machines. One of them (clnt-1) will be the client for our storage system. We can easily take its configuration as-is from the previous test environment.

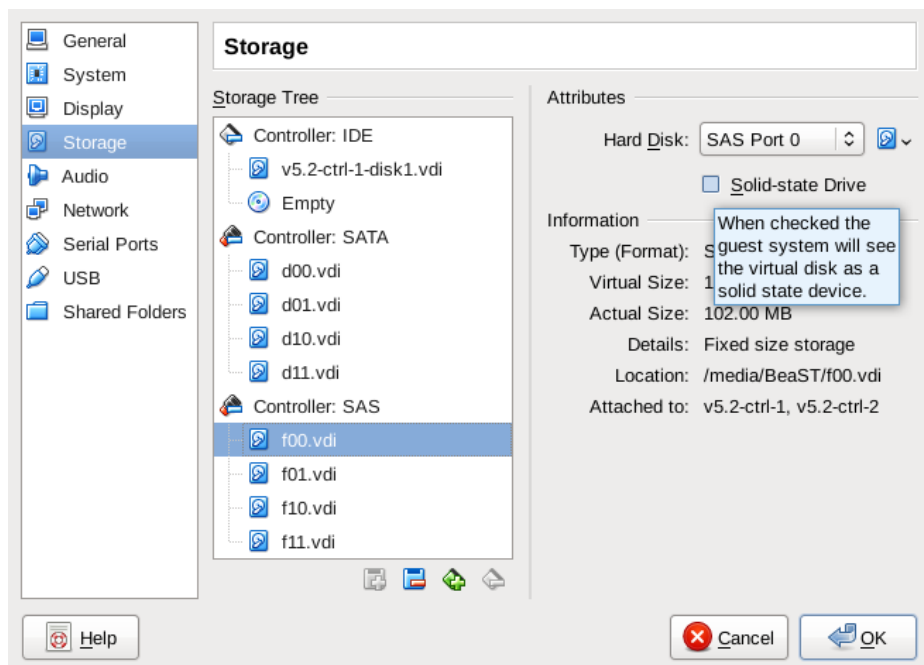
The last two machines (ctrl-a and ctrl-b) will serve as the storage controllers. These machines must be configured with at least 2,048 MB of memory to run all our tests with ZFS without issues.

With the help of the VirtualBox Virtual Media Manager, we should configure and attach to both storage controllers four fixed-sized shareable drives (d00, d01, d10, d11) for ZFS data volumes and four fixed-sized shareable drives (f00, f01, f10, f11) which we will use for ZFS cache.

We do not have any real hardware for our tests, so let's pretend that the data-drives are taken from the shared SATA shelf and imagine our cache-drives are fast Solid State Drives (SSDs) inserted into the SAS enclosure. Therefore, do not forget to add appropriate controllers to the

virtual-hosts configurations. Bear in mind, while the data-drives can be any size you prefer, the cache-drives must be at least 64 MB each.

Although it is not really mandatory for our purposes, you can even check the “Solid State Drive” option for the cache-drives to assure yourself everything is done the best way possible.



See Figure 1 with the screenshot of Oracle VM VirtualBox Manager GUI showing the storage configuration section of the ctrl-1 machine.

Figure 1. Controller storage layout example.

The network configuration is not changed. We will use two LAN connections: “private” for inter-controller and “public” for host-to-controllers communications.

Latest FreeBSD 10.3 Release can be installed on the dynamic-sized drives of all three virtual machines.

Configuration summary is shown in the table below:

Description	ctrl-a	ctrl-b	clnt-1
Inter-controller (private) network. Host-only adapter (vboxnet0)	IP: 192.168.56.10 Mask: 255.255.255.0	IP: 192.168.56.11 Mask: 255.255.255.0	
Public network. Host-only adapter (vboxnet1)	IP: 192.168.55.10 Mask: 255.255.255.0	IP: 192.168.55.11 Mask: 255.255.255.0	IP: 192.168.55.20 Mask: 255.255.255.0

Base memory	2048 MB or more	2048 MB or more	Any appropriate value starting with 512 MB will do
Shareable, fixed-sized virtual drives for ZFS data volumes on the SATA controller.	d00, d01, d10, d11 – each drive is 100 MB size or more	d00, d01, d10, d11 – each drive is 100 MB or more	
Shareable, fixed-sized virtual drives for ZFS cache on SAS controller	f00, f01, f10, f11 – at least 64 MB each	f00, f01, f10, f11 – at least 64 MB each	
System virtual drives (Dynamic-sized) on the IDE controller	At least 5 GB to store FreeBSD 10.3-Release default installation	At least 5 GB to store FreeBSD 10.3-Release default installation	At least 5 GB to store FreeBSD 10.3-Release default installation

Install FreeBSD on the ctrl-a and the ctrl-b virtual machines using default parameters and ada0 (dynamic-sized drive on the IDE controller) as the drive for the root file system. Then configure general parameters in /etc/rc.conf. This file for each controller can be easily taken from the previous environment:

ctrl-a	ctrl-b
hostname="ctrl-a"	hostname="ctrl-b"
ifconfig_em0="inet 192.168.56.10 netmask 255.255.255.0" # Inter-controller LAN	ifconfig_em0="inet 192.168.56.11 netmask 255.255.255.0" # Inter-controller LAN
ifconfig_em1="inet 192.168.55.10 netmask 255.255.255.0" # Public network	ifconfig_em1="inet 192.168.55.11 netmask 255.255.255.0" # Public network
sshd_enable="YES"	sshd_enable="YES"
# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable	# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable
dumpdev="AUTO"	dumpdev="AUTO"
# VirtualBox guest additions	# VirtualBox guest additions
vboxguest_enable="YES"	vboxguest_enable="YES"
vboxservice_enable="YES"	vboxservice_enable="YES"
# iSCSI	# iSCSI
ctld_enable="YES" # Targets	ctld_enable="YES" # target
iscsid_enable="YES" # Initiators	iscsid_enable="YES" # initiator

Do not forget to set iSCSI “disconnection on fail” kernel variable in /etc/sysctl.conf on both systems to be able to failover in case of disaster to the alive controller:

```
kern.iscsi.fail_on_disconnection=1
```


After reboot check if everything runs well. Note that shared data-drives should be recognized by the kernel as ada1, ada2, ada3, ada4 and shared cache-drives would be da0, da1, da2, da3:

```
root@ctrl-a:/home/beast # dmesg | grep "da[0-9]:"

da0: <VBOX HARDDISK 1.0> Fixed Direct Access SPC-3 SCSI device
da0: 300.000MB/s transfers
da0: Command Queueing enabled
da0: 100MB (204800 512 byte sectors)
da1: <VBOX HARDDISK 1.0> Fixed Direct Access SPC-3 SCSI device
da1: 300.000MB/s transfers
da1: Command Queueing enabled
da1: 100MB (204800 512 byte sectors)
da2: <VBOX HARDDISK 1.0> Fixed Direct Access SPC-3 SCSI device
da2: 300.000MB/s transfers
da2: Command Queueing enabled
da2: 100MB (204800 512 byte sectors)
da3: <VBOX HARDDISK 1.0> Fixed Direct Access SPC-3 SCSI device
da3: 300.000MB/s transfers
da3: Command Queueing enabled
da3: 100MB (204800 512 byte sectors)
ada0: <VBOX HARDDISK 1.0> ATA-6 device
ada0: Serial Number VB9c2e46e9-3f3664e0
ada0: 33.300MB/s transfers (UDMA2, PIO 65536bytes)
ada0: 20480MB (41943040 512 byte sectors)
ada0: Previously was known as ad0
```

```
ada1: <VBOX HARDDISK 1.0> ATA-6 SATA 2.x device
ada1: Serial Number VB9508e565-d9dfd8c7
ada1: 300.000MB/s transfers (SATA 2.x, UDMA6, PIO 8192bytes)
ada1: Command Queueing enabled
ada1: 100MB (204800 512 byte sectors)
ada1: Previously was known as ad4

ada2: <VBOX HARDDISK 1.0> ATA-6 SATA 2.x device
ada2: Serial Number VB6a067a06-5a2a2e74
ada2: 300.000MB/s transfers (SATA 2.x, UDMA6, PIO 8192bytes)
ada2: Command Queueing enabled
ada2: 100MB (204800 512 byte sectors)
ada2: Previously was known as ad6

ada3: <VBOX HARDDISK 1.0> ATA-6 SATA 2.x device
ada3: Serial Number VB07f17028-9962c138
ada3: 300.000MB/s transfers (SATA 2.x, UDMA6, PIO 8192bytes)
ada3: Command Queueing enabled
ada3: 100MB (204800 512 byte sectors)
ada3: Previously was known as ad8

ada4: <VBOX HARDDISK 1.0> ATA-6 SATA 2.x device
ada4: Serial Number VBccf97b29-a1147aa3
ada4: 300.000MB/s transfers (SATA 2.x, UDMA6, PIO 8192bytes)
ada4: Command Queueing enabled
ada4: 100MB (204800 512 byte sectors)
ada4: Previously was known as ad10
```


ZFS pools basic configuration

If everything is done properly, we can configure ZFS pools on the controllers.

Let’s use ada1/ada2 drives to create ctrl_a_m0 pool on the ctrl-a controller and ada3/ada4 drives to form ctrl_b_m0 pool on the ctrl-b controller. Then add volumes – v0 to each pool:

ctrl-a	ctrl-b
zpool create -m none ctrl-a_m0 /dev/ada1 /dev/ada2 zfs create -V 120M ctrl-a_m0/v0	zpool create -m none ctrl-b_m0 /dev/ada3 /dev/ada4 zfs create -V 120M ctrl-b_m0/v0

The only interesting options are:

- m none - prevents occasional pool mounting.
- V - desired volume size.

To check the result, run:

```
root@ctrl-a:/home/beast # zpool status

pool: ctrl_a_m0

state: ONLINE

scan: none requested

config:

    NAME            STATE             READ  WRITE CKSUM
    ctrl-a_m0        ONLINE            0     0     0
    ada1             ONLINE            0     0     0
    ada2             ONLINE            0     0     0
```

```
errors: No known data errors

root@ctrl-a:/home/beast # zfs list

NAME                                USED  AVAIL  REFER  MOUNTPOINT
ctrl-a_m0                          124M  3.92M   19K    none
ctrl-a_m0/v0                       124M  128M    8K    -
```

Read and write caches

Now let's try to move ZFS cache to the shared devices. In our case these will be da0 – da3 drives, which we agree to consider the fast SSDs.

First, we will try to implement quazi-write cache with ZFS Intent Log (ZIL) forcing it to handle both types of synchronous and asynchronous transactions and write them to the shared drive da0 (the ctrl-a_m0 pool on the ctrl-a controller) and da2 (ctrl-b_m0 pool on on the ctrl-b):

ctrl-a	ctrl-b
<pre>zpool add ctrl-a_m0 log /dev/da0 # Always write and flush all file system transactions. zfs set sync=always ctrl-a_m0</pre>	<pre>zpool add ctrl-b_m0 log /dev/da2 # Always write and flush all file system transactions. zfs set sync=always ctrl-b_m0</pre>

Second, add the shared drives to the pools: da1 (on the ctrl-a) and da3 (on the ctrl-b) to use as the read-caches for both pools. Finally, disable in-memory read-cache completely and move it to the shared devices.

ARC / L2ARC Configuration (read cache):

ctrl-a	ctrl-b
<pre>zpool add ctrl-a_m0 cache /dev/da1 # Turn-off ARC caching zfs set primarycache=none ctrl-a_m0 # Enable L2ARC caching to the shared device zfs set secondarycache=all ctrl-a_m0</pre>	<pre>zpool add ctrl-b_m0 cache /dev/da3 # Turn-off ARC caching on ctrl-b zfs set primarycache=none ctrl-b_m0 # Enable L2ARC caching to the shared device zfs set secondarycache=all ctrl-b_m0</pre>

ZFS pools final configurations

On the last step of ZFS configuration we should import pools from opposite controllers. Then we have to set “failmode=continue” property.

This is quite dangerous because we just reject to stop working even if any errors are detected on the pools, but it is necessary to be able to failover to the next controller in case of disaster. So:

ctrl-a	ctrl-b
<pre>zpool import -N ctrl-b_m0 zpool set failmode=continue ctrl-a_m0 zpool set failmode=continue ctrl-b_m0</pre>	<pre>zpool import -N ctrl-a_m0 zpool set failmode=continue ctrl-a_m0 zpool set failmode=continue ctrl-b_m0</pre>

-N option of “zpool import” command above prevents the pool from being mounted.

Finally, check zpool configuration. We should be able to see both pools on both storage controllers:

```
root@ctrl-a:/home/beast # zpool status

pool: ctrl-a_m0

state: ONLINE

scan: none requested

config:
```

NAME	STATE	READ	WRITE	CKSUM
ctrl-a_m0	ONLINE	0	0	0
ada1	ONLINE	0	0	0
ada2	ONLINE	0	0	0
logs				
da0	ONLINE	0	0	0
cache				
da1	ONLINE	0	0	0

errors: No known data errors

pool: ctrl-b_m0

state: ONLINE

scan: none requested

config:

NAME	STATE	READ	WRITE	CKSUM
ctrl-b_m0	ONLINE	0	0	0
ada3	ONLINE	0	0	0
ada4	ONLINE	0	0	0
logs				
da2	ONLINE	0	0	0
cache				
da3	ONLINE	0	0	0

errors: No known data errors

The Arbitrator

The arbitrator mechanism is similar to the one that was discussed in the previous paper. The only difference is that now we use data-volumes on ZFS pools. So just edit /etc/ctl.conf to add appropriate inter-controller communication configurations for the v0 volumes:

ctrl-a	ctrl-b
<pre>portal-group pg0 { discovery-auth-group no-authentication listen 192.168.56.10 } target iqn.2016-01.local.sss.private:target0 { auth-group no-authentication portal-group pg0 lun 0 { path /dev/zvol/ctrl-a_m0/v0 } }</pre>	<pre>portal-group pg0 { discovery-auth-group no-authentication listen 192.168.56.11 } target iqn.2016-01.local.sss.private:target0 { auth-group no-authentication portal-group pg0 lun 0 { path /dev/zvol/ctrl-b_m0/v0 } }</pre>

Then restart ctld daemon on both controllers to update iSCSI targets:

ctrl-a	ctrl-b
<pre>service ctld restart</pre>	<pre>service ctld restart</pre>

From both controllers connect with the LUNs on the opposite controllers:

ctrl-a	ctrl-b
<pre>iscsictl -A -p 192.168.56.11 -t iqn. 2016-01.local.sss.private:target0</pre>	<pre>iscsictl -A -p 192.168.56.10 -t iqn. 2016-01.local.sss.private:target0</pre>

And finally, assemble the arbitrating construction by creating active-passive multipath devices, which have active paths pointing to the opposite controllers at their initial state:

ctrl-a	ctrl-b
gmultipath create CTRL_B_BACK /dev/da4 /dev/zvol/ctrl-b_m0/v0	gmultipath create CTRL_A_BACK /dev/da4 /dev/zvol/ctrl-a_m0/v0

Front-end configuration

The tandem of the arbitrator and the external shared cache forms a reliable structure resistant to a single controller failure. Therefore we can continue to configure host-connection configuration and update /etc/ctl.conf with the “public” sections to allow host access:

ctrl-a	ctrl-b
<pre>portal-group pg0 { discovery-auth-group no-authentication listen 192.168.56.10 } portal-group pg1 { discovery-auth-group no-authentication listen 192.168.55.10 } target iqn.2016-01.local.sss.private:target0 { auth-group no-authentication portal-group pg0 lun 0 { path /dev/zvol/ctrl-a_m0/v0 } }</pre>	<pre>portal-group pg0 { discovery-auth-group no-authentication listen 192.168.56.11 } portal-group pg1 { discovery-auth-group no-authentication listen 192.168.55.11 } target iqn.2016-01.local.sss.private:target0 { auth-group no-authentication portal-group pg0 lun 0 { path /dev/zvol/ctrl-b_m0/v0 } }</pre>

ctrl-a	ctrl-b
<pre>target iqn.2016-01.local.sss.public:target0 { auth-group no-authentication portal-group pg1 lun 0 { path /dev/zvol/ctrl-a_m0/v0 } lun 1 { path /dev/multipath/CTRL_B_BACK } }</pre>	<pre>target iqn.2016-01.local.sss.public:target0 { auth-group no-authentication portal-group pg1 lun 0 { path /dev/zvol/ctrl-b_m0/v0 } lun 1 { path /dev/multipath/CTRL_A_BACK } }</pre>

Then force ctld daemon to re-read and refresh iSCSI targets configuration on both controllers:

ctrl-a	ctrl-b
<pre>killall -HUP ctld</pre>	<pre>killall -HUP ctld</pre>

Check for essential messages in the dmesg output to see if everything goes smooth:

```
root@ctrl-a:/home/beast # dmesg | tail -10

da4 at iscsi1 bus 0 scbus8 target 0 lun 0

da4: <FREEBSD CTLDISK 0001> Fixed Direct Access SPC-4 SCSI device

da4: Serial Number MYSERIAL    0

da4: 150.000MB/s transfers

da4: Command Queueing enabled

da4: 120MB (245760 512 byte sectors)
```

```

GEOM_MULTIPATH: CTRL_B_BACK created
GEOM_MULTIPATH: da4 added to CTRL_B_BACK
GEOM_MULTIPATH: da4 is now active path in CTRL_B_BACK
GEOM_MULTIPATH: zvol/ctrl-b_m0/v0 added to CTRL_B_BACK

```

Figure 2 shows our storage system architecture layout we just created:

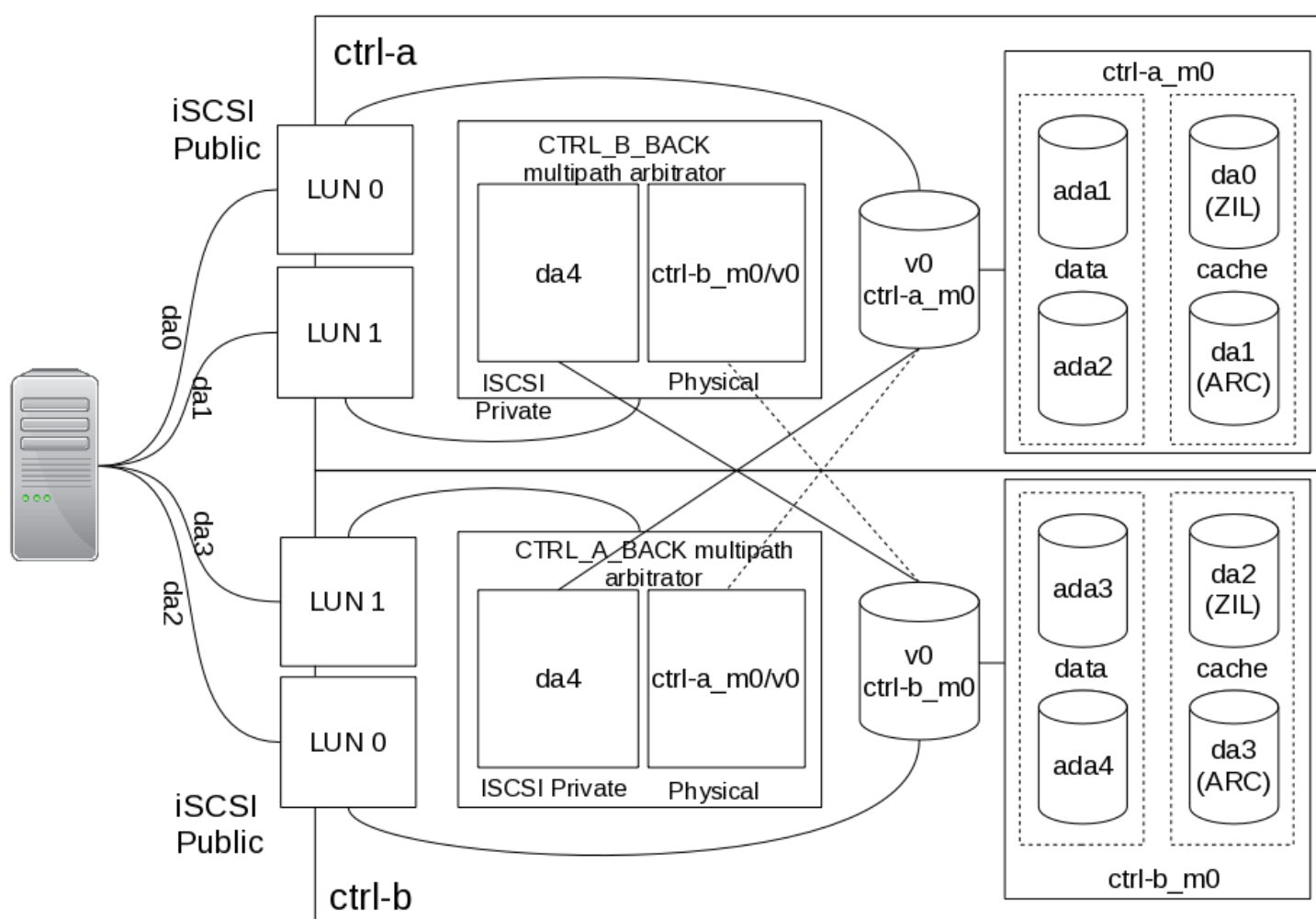


Figure 2. Dual-controller storage architecture with ZFS and external cache overview.

The client side

As the storage controllers are ready to serve requests, we can prepare the client. The **clnt-1** initial configuration is identical to the previous environment and can be used as-is. The essential lines of **/etc/rc.conf** are shown below:

```

hostname="clnt-1"

ifconfig_em0="inet 192.168.55.20 netmask 255.255.255.0" # Public net-
work

```



```

sshd_enable="YES"

# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable
dumpdev="AUTO"

# VirtualBox guest additions
vboxguest_enable="YES"
vboxservice_enable="YES"

# iSCSI

iscsid_enable="YES"      # Initiators

```

The `/etc/sysctl.conf` file is also the same:

```
kern.iscsi.fail_on_disconnection=1
```

After the basic client preparation we can try to connect with public iSCSI targets of both storage controllers:

```

iscsictl -A -p 192.168.55.10 -t iqn.2016-01.local.sss.public:target0
iscsictl -A -p 192.168.55.11 -t iqn.2016-01.local.sss.public:target0

```

Check if `dmesg` output shows the appearance of the new `da0`, `da1`, `da2`, and `da3` drives:

```

root@clnt-1:/home/beast # dmesg | grep "^da[0-9]:"
da0: <FREEBSD CTLDISK 0001> Fixed Direct Access SPC-4 SCSI device

```

```
da0: Serial Number MYSERIAL    1
da0: 150.000MB/s transfers
da0: Command Queueing enabled
da0: 120MB (245760 512 byte sectors: 64H 32S/T 120C)
da1: <FREEBSD CTLDISK 0001> Fixed Direct Access SPC-4 SCSI device
da1: Serial Number MYSERIAL    2
da1: 150.000MB/s transfers
da1: Command Queueing enabled
da1: 120MB (245760 512 byte sectors: 64H 32S/T 120C)
da2: <FREEBSD CTLDISK 0001> Fixed Direct Access SPC-4 SCSI device
da2: Serial Number MYSERIAL    1
da2: 150.000MB/s transfers
da2: Command Queueing enabled
da2: 120MB (245760 512 byte sectors: 64H 32S/T 120C)
da3: <FREEBSD CTLDISK 0001> Fixed Direct Access SPC-4 SCSI device
da3: Serial Number MYSERIAL    2
da3: 150.000MB/s transfers
da3: Command Queueing enabled
da3: 120MB (245760 512 byte sectors: 64H 32S/T 120C)
```

Then create appropriate multipathing devices to access both storage controllers:

```
gmultipath create CTRL_A /dev/da0 /dev/da3
gmultipath create CTRL_B /dev/da2 /dev/da1
```

Finally we can create a striped volume, then add and mount a filesystem:

```
gstripe create ZDATA /dev/multipath/CTRL_A /dev/multipath/CTRL_B
newfs /dev/stripe/ZDATA
mount /dev/stripe/ZDATA /storage
```

If everything is well, we will see the appearance of the new mounted filesystem:

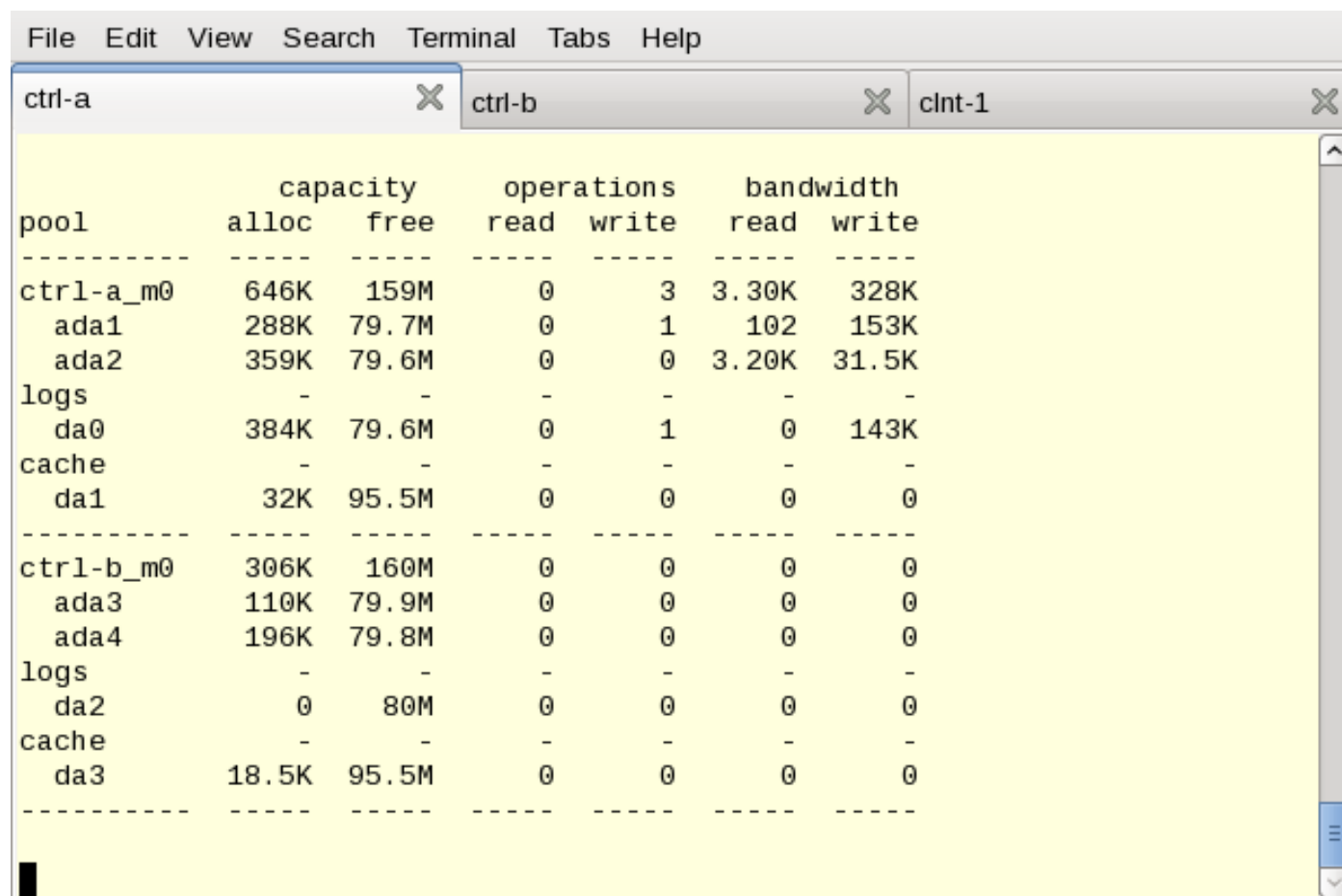
```
root@clnt-1:/home/beast # df -h | grep ZDATA

/dev/stripe/ZDATA      232M      8.0K      213M      0%      /storage
```

Then we can run all the tests we got familiar with in the previous version of the BeaST system. So let's start with a file copy operation:

```
root@clnt-1:/home/beast # cp ports.tgz /storage/
```

Figures 3, 4, and 5 show us the state of both controllers and the client:



pool	capacity		operations		bandwidth	
	alloc	free	read	write	read	write
ctrl-a_m0	646K	159M	0	3	3.30K	328K
ada1	288K	79.7M	0	1	102	153K
ada2	359K	79.6M	0	0	3.20K	31.5K
logs	-	-	-	-	-	-
da0	384K	79.6M	0	1	0	143K
cache	-	-	-	-	-	-
da1	32K	95.5M	0	0	0	0
ctrl-b_m0	306K	160M	0	0	0	0
ada3	110K	79.9M	0	0	0	0
ada4	196K	79.8M	0	0	0	0
logs	-	-	-	-	-	-
da2	0	80M	0	0	0	0
cache	-	-	-	-	-	-
da3	18.5K	95.5M	0	0	0	0

Figure 3. The ctrl-a normal operations.

You can see “zpool iostat -v 5” on Figure 3, which shows us the activity on ctrl-a_m0 pool detected by the ctrl-a. At the same time workload of the ctrl-b_m0 pool is hidden to the ctrl-a controller. Actually both pools are utilized but from different controllers.

pool	capacity		operations		bandwidth	
	alloc	free	read	write	read	write
ctrl-a_m0	284K	160M	0	0	0	0
ada1	108K	79.9M	0	0	0	0
ada2	176K	79.8M	0	0	0	0
logs	-	-	-	-	-	-
da0	0	80M	0	0	0	0
cache	-	-	-	-	-	-
da1	18K	95.5M	0	0	0	0
ctrl-b_m0	4.59M	155M	0	2	0	101K
ada3	2.30M	77.7M	0	0	0	610
ada4	2.29M	77.7M	0	0	0	406
logs	-	-	-	-	-	-
da2	6.25M	73.8M	0	1	0	100K
cache	-	-	-	-	-	-
da3	32K	95.5M	0	0	0	305

The same situation on the ctrl_b controller (Figure 4): the ctrl-b_m0 pool activity is shown, but the ctrl-a_m0 workload is not detected.

Figure 4. The ctrl-b normal operations.

From the client side (Figure 5) both primary paths are active and the data goes normally though both owner controllers.

pool	capacity		operations		bandwidth	
	alloc	free	read	write	read	write
ctrl-a_m0	284K	160M	0	0	0	0
ada1	108K	79.9M	0	0	0	0
ada2	176K	79.8M	0	0	0	0
logs	-	-	-	-	-	-
da0	0	80M	0	0	0	0
cache	-	-	-	-	-	-
da1	18K	95.5M	0	0	0	0
ctrl-b_m0	4.59M	155M	0	2	0	101K
ada3	2.30M	77.7M	0	0	0	610
ada4	2.29M	77.7M	0	0	0	406
logs	-	-	-	-	-	-
da2	6.25M	73.8M	0	1	0	100K
cache	-	-	-	-	-	-
da3	32K	95.5M	0	0	0	305

Now let's actually fail one of the controllers and see the result. Traditionally I will kick-off the ctrl-a.

Figure 4. The ctrl-b normal operations.

Figure 6 now shows the survived controller:

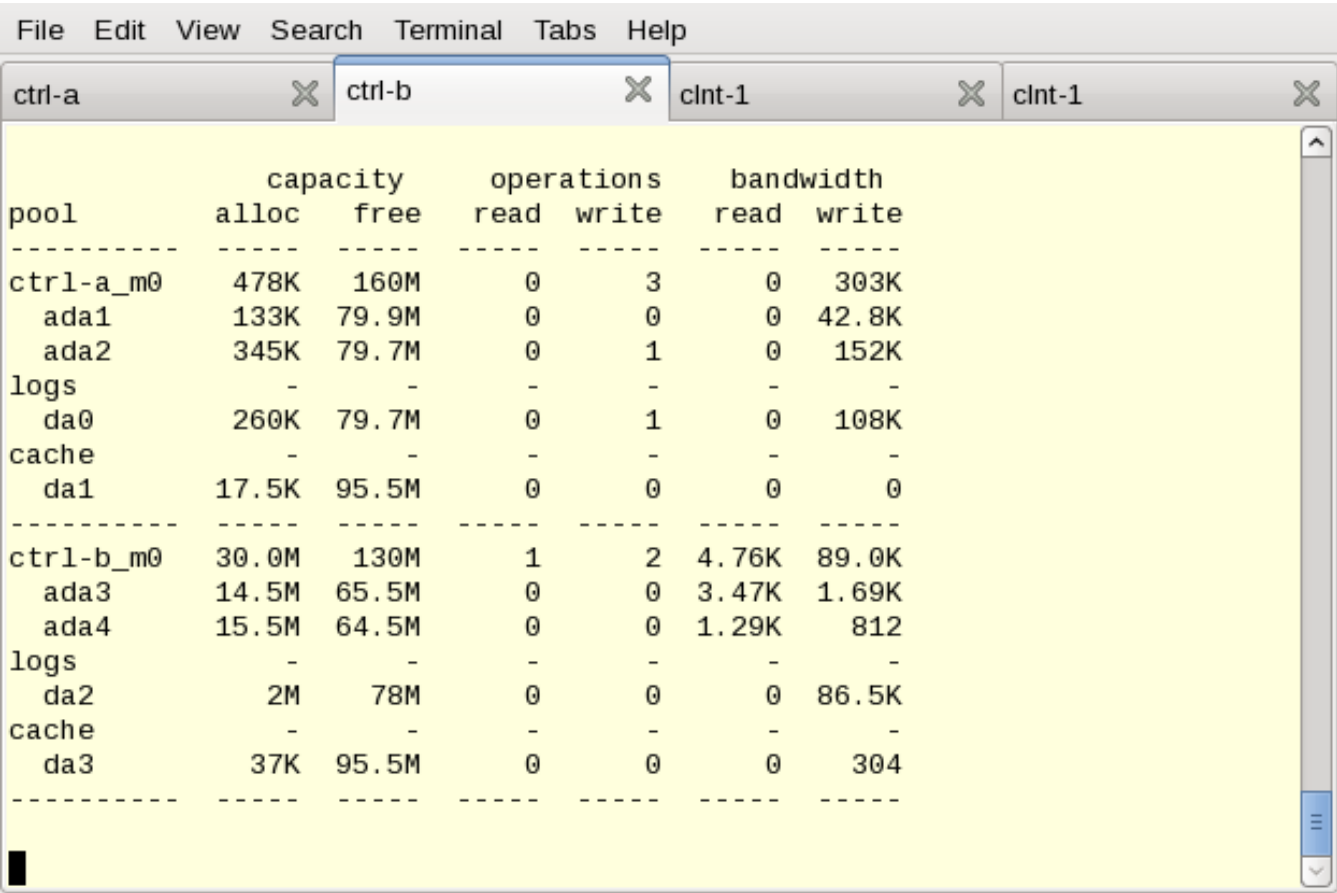


Figure 6. The ctrl-b after ctrl-a failed.

Now the ctrl-b takes all the clients workload: the arbitrator directed all data traffic through the survived controller.

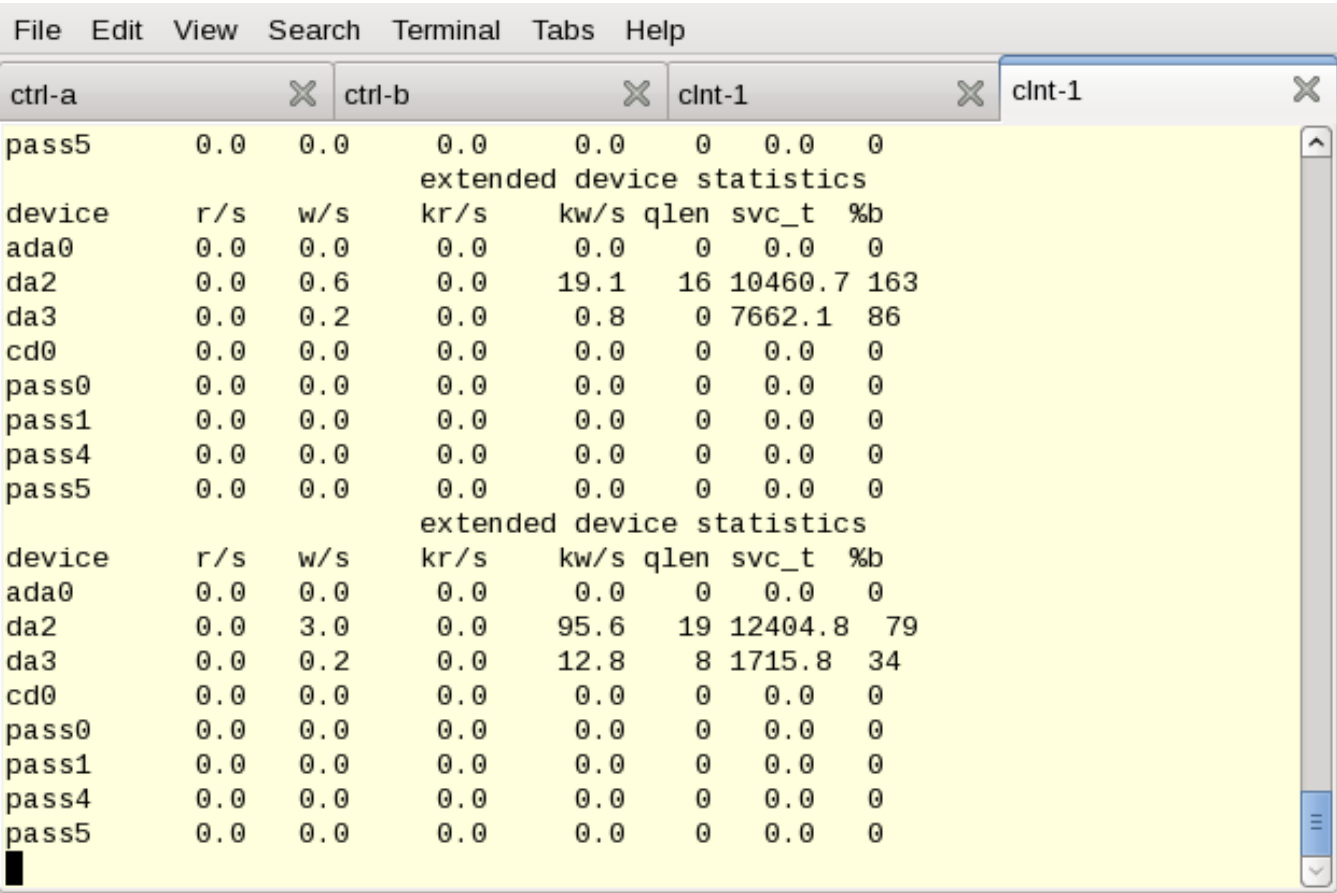


Figure 7. The client clnt-1 after ctrl-a failure.

And it is not really a surprise as the clnt-1 has lost all paths to the ctrl-a. On Figure 7 you can see that da0 and da1 have disappeared while da2/da3 is taking all the workload.

After the file-copy operation has been finished, we can check if the data are written correctly:

```
root@clnt-1:/home/beast # md5 ports.tgz

MD5 (ports.tgz) = 82a5d6a7a3a89b7a5185a543fa6b3a56

root@clnt-1:/home/beast # md5 /storage/ports.tgz

MD5 (/storage/ports.tgz) = 82a5d6a7a3a89b7a5185a543fa6b3a56
```

We can state that the system works well in the laboratory. But now we are on a very slippery floor, as ZFS is not a cluster system and therefore is not designed to be run in the shared environments. Our storage solution is still for testing purposes only. So nobody can guarantee that everything will work in production. Beware of data loss!

The recovery procedure

From any point of view this is definitely not the brightest side of life. Actually, it is painful because the most important thing we should remember now, is that we will lose data if the last survived controller drops its access to the pools. In other words, the last controller must stay online until the recovery is finished: no shutdowns at all, even planned one!

Let's study the case. First of all, we must check the pool status:

```
root@ctrl-b:/home/beast # zpool status

pool: ctrl-a_m0

state: ONLINE

scan: none requested

config:

   NAME                STATE          READ  WRITE CKSUM
   ctrl-a_m0            ONLINE         0     0     0
   ada1                 ONLINE         0     0     0
```



```
    ada2      ONLINE      0      0      0
logs
    da0      ONLINE      0      0      0
cache
    da1      ONLINE      0      0      0
errors: No known data errors
```

```
pool: ctrl-b_m0
state: ONLINE
scan: none requested
config:
```

```
NAME          STATE      READ WRITE CKSUM
ctrl-b_m0     ONLINE      0     0     0
    ada3     ONLINE      0     0     0
    ada4     ONLINE      0     0     0
logs
    da2     ONLINE      0     0     0
cache
    da3     ONLINE      0     0     0
errors: No known data errors
```

But we must not trust it at all! As we remember, one controller cannot see pool activities of the other controller. Additionally, we disabled error detection on the pools by the “failmode=continue” option with our own hands.

So let’s run the “scrub” procedure on the pool attached to the failed controller, then check the status to see the real picture of disaster:

```
root@ctrl-b:/home/beast # zpool scrub ctrl-a_m0

root@ctrl-b:/home/beast # zpool status -v

pool: ctrl-a_m0

state: ONLINE

status: One or more devices has experienced an error resulting in
data
    corruption.  Applications may be affected.

action: Restore the file in question if possible.  Otherwise restore
the
    entire pool from backup.

see: http://illumos.org/msg/ZFS-8000-8A

scan: scrub repaired 1K in 0h0m with 2 errors on Tue Apr 19
16:07:39 2016

config:

    NAME                STATE          READ  WRITE CKSUM
    ctrl-a_m0            ONLINE         0     0     2
    ada1                 ONLINE         0     0     4
    ada2                 ONLINE         0     0    10
    logs
    da0                  ONLINE         0     0     0
    cache
```

```
da1      ONLINE      0      0      0

errors: Permanent errors have been detected in the following files:
<metadata>:<0x1b>
      <metadata>:<0x20>

pool:  ctrl-b_m0
state:  ONLINE
      scan: none requested
config:

NAME      STATE      READ  WRITE  CKSUM
ctrl-b_m0  ONLINE      0      0      0
  ada3     ONLINE      0      0      0
  ada4     ONLINE      0      0      0
logs
  da2      ONLINE      0      0      0
cache
  da3      ONLINE      0      0      0

errors: No known data errors
```

Finally, run scrub on the ctrl-b_m0 pool and try to boot the ctrl-a controller.

Immediately after the ctrl-a is booted, login to its console and run “zpool export” to disconnect the controller from both of the pools:

```
root@ctrl-a:/home/beast # zpool export ctrl-a_m0
root@ctrl-a:/home/beast # zpool export ctrl-b_m0
```

Now we can breath normally. Calmly reboot ctrl-a once more, then import pools back again:

```
root@ctrl-a:/home/beast # zpool import -N ctrl-a_m0
root@ctrl-a:/home/beast # zpool import -N ctrl-b_m0
```

Then run scrub on both controllers:

```
root@ctrl-a:/home/beast # zpool scrub ctrl-a_m0
root@ctrl-a:/home/beast # zpool scrub ctrl-b_m0
root@ctrl-b:/home/beast # zpool scrub ctrl-a_m0
root@ctrl-b:/home/beast # zpool scrub ctrl-b_m0
```

Also you may need to clear errors on the pools:

```
zpool clear ctrl-a_m0
zpool clear ctrl-b_m0
```

If we succeed with the pools recovery procedure we should carefully restore the arbitrator, all broken iSCSI connections, and multipathing primary paths. In other words we have to repeat most of the steps specified in the article once more. It is a very tedious process, but it should be done carefully to finish the storage system restoration.

What is next

We have added ZFS to the BeaST project and that is very good. But bearing in mind the real life and performance demands of the different workloads, it is probably not the best idea to put the main system cache to the external solid state drives. Better if we are able to implement in-memory cache and make it somehow to be mirrored between both controllers. This would be the first task for the future development.

Second, we still must test the BeaST project under the pressure of high workloads on the real hardware and test not only the performance but the stability of our storage concept.

Third, FreeBSD 10.3 Release brought to the table the high-availability options for the CAM Target Layer and we should check how the BeaST can benefit from the new features.



About the Author:

I am a proud SAN/storage IBMer. 10 years of experience in large SAN and storage environments: mainly Hitachi, HP and Brocade. Empty – expect-like tool author. FreeBSD enthusiast.

Mikhail E. Zakharov, zmey20000@yahoo.com

FREENAS MINI STORAGE APPLIANCE

IT SAVES YOUR LIFE.



HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**



Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time.***

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured



<http://www.iXsystems.com/mini>



FREENAS CERTIFIED STORAGE



With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

- » Custom built and optimized for your use case
- » Installed, configured, tested, and guaranteed to work out of the box
- » Supported by the Silicon Valley team that designed and built it
- » Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

<http://www.ixsystems.com/storage/freenas-certified-storage/>



Secure VPNs with Gre and Strongswan for Small Business Networks with FreeBSD

by Antonio Francesco Gentile

Communication is a competitive advantage for any company. With secure, reliable, and confidential communication it is possible to reduce costs, and improve the efficiency of production processes. In this article we will see how to implement them in FreeBSD.

What you will learn...

In this paper we will learn to setup and manage secure IPSEC over Gre Tunnels.

What you should know...

Basic BSD Networking Setup, basic Networking structure knowledge, and basic Network security concepts.

Startup

A VPN (Virtual Private Network) is used when one needs to create a link between two or more private networks over a public network (like the Internet). Once the connection is established between the two private networks, users will see the counterpart network in a completely transparent way, as if they were physically connected to each other. But you have to keep in mind that the maximum connection speed between the two networks is defined by the public network, and not from the standard 100/1000 mbit LAN. Another very important characteristic of the VPN is to create a secure communication system; this, so you can be confident of security even in the case of transferring confidential and sensitive data.

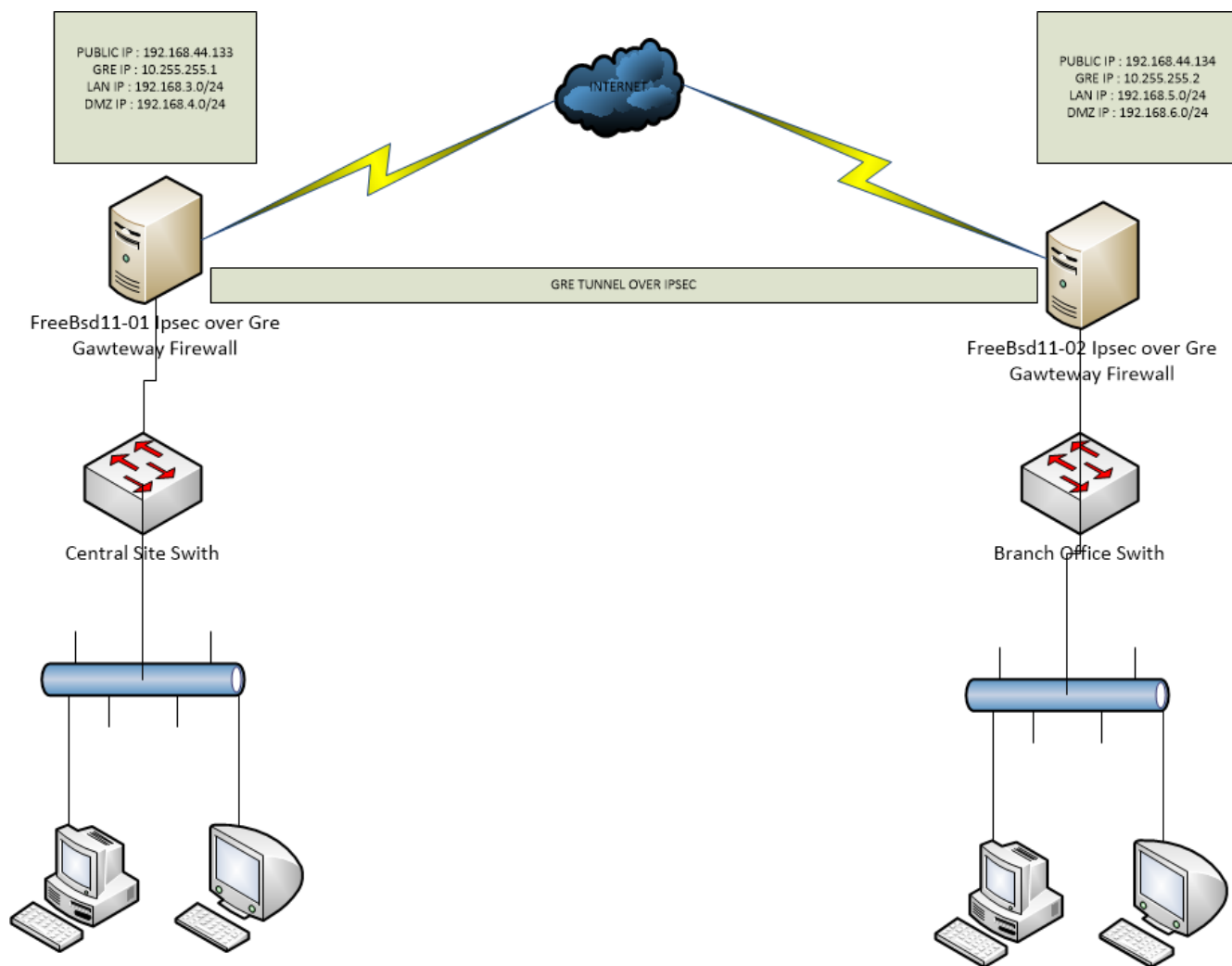


Figure 1. A typical Small Business LAN to LAN Ipsec over Gre Scenario

Summing up a VPN is "private" because it offers a nearly identical service to a private LAN, and "virtual" because in reality the data stream passes from the public network (Internet). It serves:

1. For companies to connect peripheral headquarters without the need for dedicated connections.
2. Two common users to share resources through the Internet.
3. For an admin to administer a remote machine.
4. To make a confidential shared network (WLAN or LAN of hub).

All traffic between the two endpoints of the VPN is encapsulated in pre-tunnel. The tunnels can be on different levels of the ISO/OSI stack:

1. IPSEC (a layer 3)
2. PPTP (a layer 3)
3. OpenVPN, I2tp, vtund, cipe, etc. (A layer 4)

The implementation presented in this article will use the GRE tunnel over IPSEC Gateway Firewall FreeBSD 11.

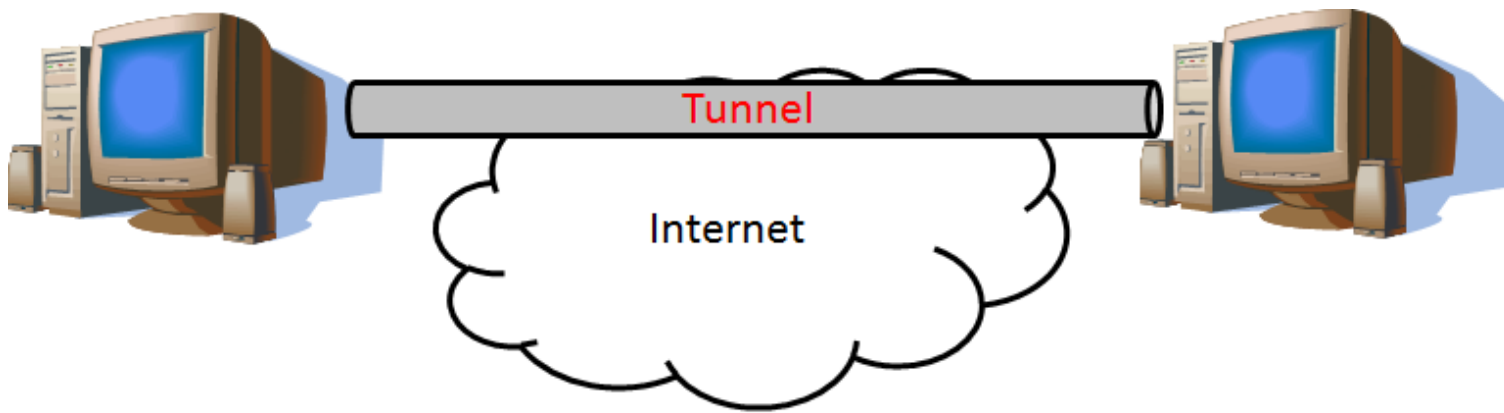


Figure 2. *Gre Tunnel over the Internet.*

Our Scenario:

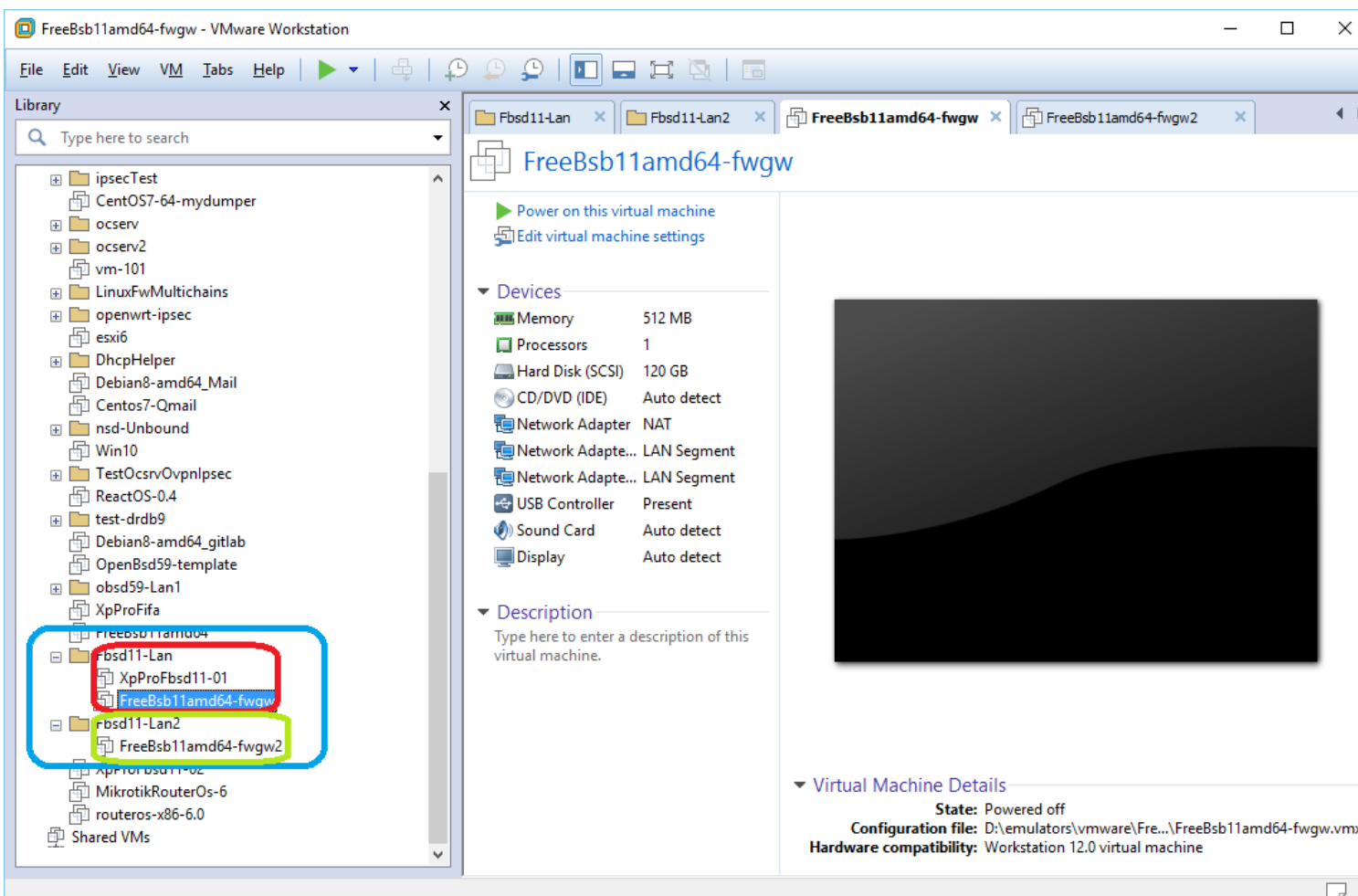


Figure 3. *Vmware Workstation Setup of two remote networks connected over the Internet.*

The proposed scenario was implemented on a real gateway and tested on VMware architecture before it goes into production. Here is the network parameters used and the appropriate configuration file that was necessary to edit:

```
/etc/rc.conf  
  
/boot/loader.conf
```

```
/usr/local/etc/ipsec.conf  
  
/usr/local/etc/ipfw.rules  
  
/usr/local/etc/pf.conf  
  
/etc/ipsec.secrets ( symbolic link to /usr/local/etc/ipsec.secrets)
```

<u>FIRST GATEWAY IPSEC</u>	<u>SECOND GATEWAY IPSEC</u>
<i>PUBLIC IP : 192.168.44.133</i>	<i>PUBLIC IP : 192.168.44.134</i>
<i>GRE IP : 10.255.255.1</i>	<i>GRE IP : 10.255.255.2</i>
<i>LAN IP : 192.168.3.0/24</i>	<i>LAN IP : 192.168.5.0/24</i>

VPN Setup in Practice

An overlay VPN is a network of computers connected together that generate a given topology. The peculiarity of these networks is that they are not physical, but rather logical, and for this reason are made above other existing networks, in our case the INTERNET. The fundamental prerequisite for our project is that each node belonging to the overlay, or wishing to join this network, has a public IP address through which the management software may be placed in direct communication. The first problem to be overcome is the logical interconnection between network nodes. In this regard, tunneling technologies come to the rescue.

Tunneling is a technique that inserts a given protocol, in another protocol, at the same or higher level. The centerpiece of the whole project with this particular technique is to create virtual interfaces on nodes that will make up the Overlay Network. Interconnection Point-to-Point between virtual interfaces on different nodes allows us to create the Overlay Network. GRE is the perfect candidate as a tunneling protocol.

GRE (Generic Routing Encapsulation)

GRE is already integrated in most Unix-like distributions and is very simple to setup. The decision to adopt this particular type of tunneling protocol is because GRE is able to encapsulate the inside, also the broadcast and multicast traffic, used by common routing protocols.

Structure of GRE Tunneling

The nodes are physically interconnected thanks to the INTERNET network.

Security of data transmission is provided by IPsec in Transport Mode. This protocol is completely transparent to the application layer and is unable to implement encryption and authentication of IP connections. The Transport Mode it is made safe for only the IP packet payload. The IP header is removed and the payload is encrypted (ESP) and is applied to the security header (ESP/AH). Finally the original IP header is reapplied. To make this mechanism, configure the SPD (Security Policy Database) through which a SAD (Security Association Database) will then be generated.

1. SP - Rule who should be treated by a specific SA.
2. SA - it defines how to create a secure connection between two hosts.

The connection is established and data of any type can be exchanged between all network nodes including routing tables to reach the new node.

Pre-installation steps:

If one prefers, instead of loading the appropriate modules in the loader.conf file it is possible to re-compile the kernel to support IPSEC and Gre.

IPsec over GRE

(GRE encapsulated packet is sent over the internet, routing updates and route information are exchanged in clear text. The interesting traffic defined for IPsec encryption is the actual payload which does not include the GRE traffic, so ONLY the underlying payload is encrypted.)

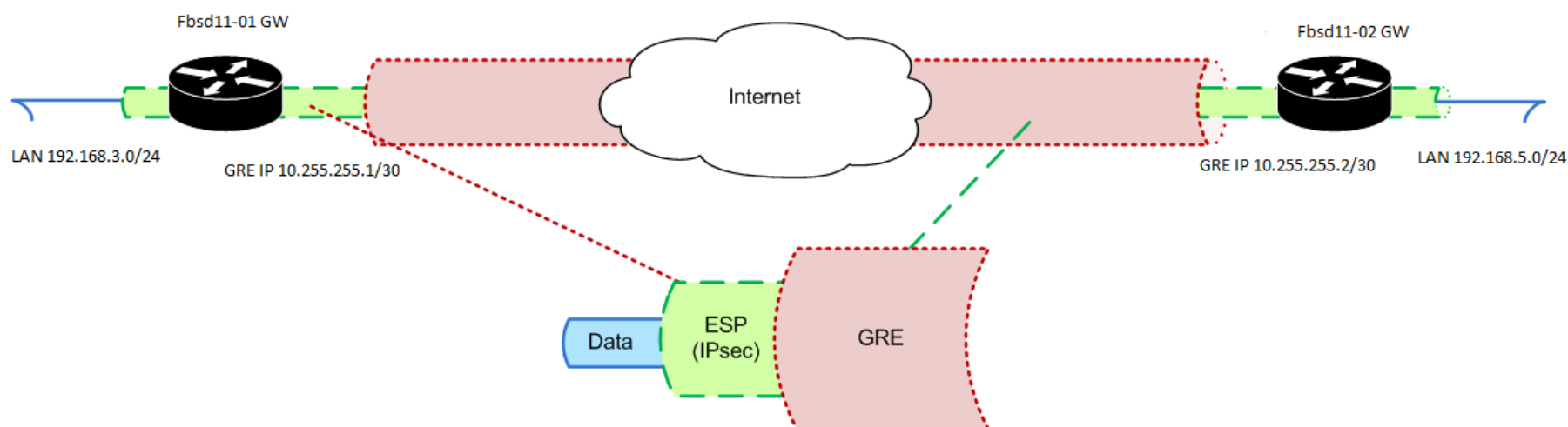


Figure 4. Operating diagram of gre over ipsec.

Prepare FreeBSD kernel:

The generic FreeBSD kernel does not come with IPsec support. So you will have to compile your own kernel. Fortunately, starting with FreeBSD 8, the NAT Traversal patch is included in the kernel sources, so you do not have to apply any patches yourself - if you need that feature. Let's start by customizing our new kernel configuration.

The standard naming convention for kernel configuration files is the name of the kernel in all caps and our new configuration will be called FBSD11IPSEC. Kernel configuration files live inside the `/usr/src/sys/architecture/conf` directory; the architecture used in this configuration will be AMD64.

Change to the configuration directory.

```
#cd /usr/src/sys/amd64/conf
```

Copy the GENERIC kernel configuration file with our name “FBSD11IPSEC” and open it for editing using your favorite text editor.

```
#copy GENERIC FBSD11IPSEC  
  
#emacs FBSD11IPSEC
```

You can find the FBSD11IPSEC configuration located here. Change the line starting with ident:

```
ident FBSD11IPSEC
```

To enable IPsec and gre one will need to add the following options to kernel configuration file:

```
options      IPSEC  
  
device       crypto  
  
device gre
```

Building and installing the new Kernel

Now we will begin the kernel recompilation. Change back to the `/usr/src` directory and issue a `make buildkernel` utilizing your new configuration file. The build kernel will take some time depending on your hardware. Once the kernel recompilation has finished, it is time to begin the install and reboot the system:

```
cd /usr/src  
  
make buildkernel KERNCONF=FBSD11IPSEC  
  
make installkernel KERNCONF=FBSD11IPSEC  
  
shutdown -r now
```

The server should now begin to shut down its currently running services, sync its disks, and reboot into new kernel. One can check that new kernel config. is being used with the following command:

```
sysctl kern.conftxt | grep ident
```

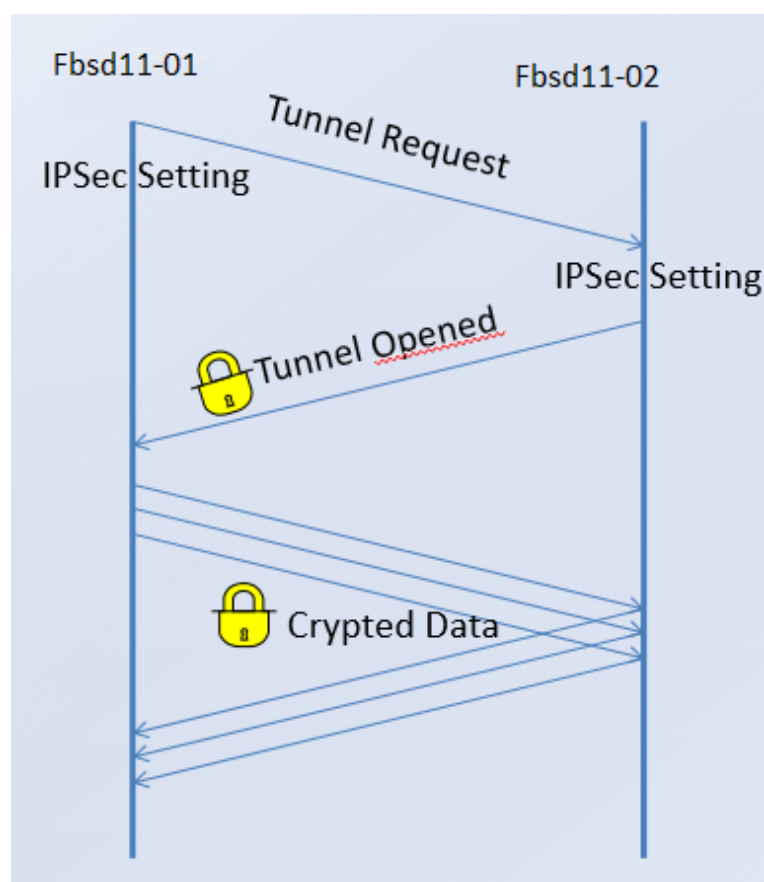
The output should be:

```
ident      FBSD11IPSEC
```

One can verify that the kernel has IPsec support using the following command, which should print a list of the ipsec specific kernel state.

```
/sbin/sysctl -a | grep ipsec
```

Installation of StrongSwan with FreeBSD Port/Package



Historically, it made use of ipsec-tools suite to encapsulate the gre traffic with ipsec, with the “Racoon” service. There are now more modern and flexible services available on both Linux and FreeBSD, as StrongSwan will be used to make data on the tunnel protection.

Figure 5. Operating diagram of gre over ipsec session.

The easiest way to install strongSwan on FreeBSD is to use the security/strongswan port:

```
cd /usr/ports/security/strongswan/ && make install clean
```

or to install the binary package with// Here this is just for the example, having a “optin” to enable our custom function:

```
pkg install strongswan
```

To get it to start at boot time, add this line to /etc/rc.conf:

```
strongswan_enable="YES"

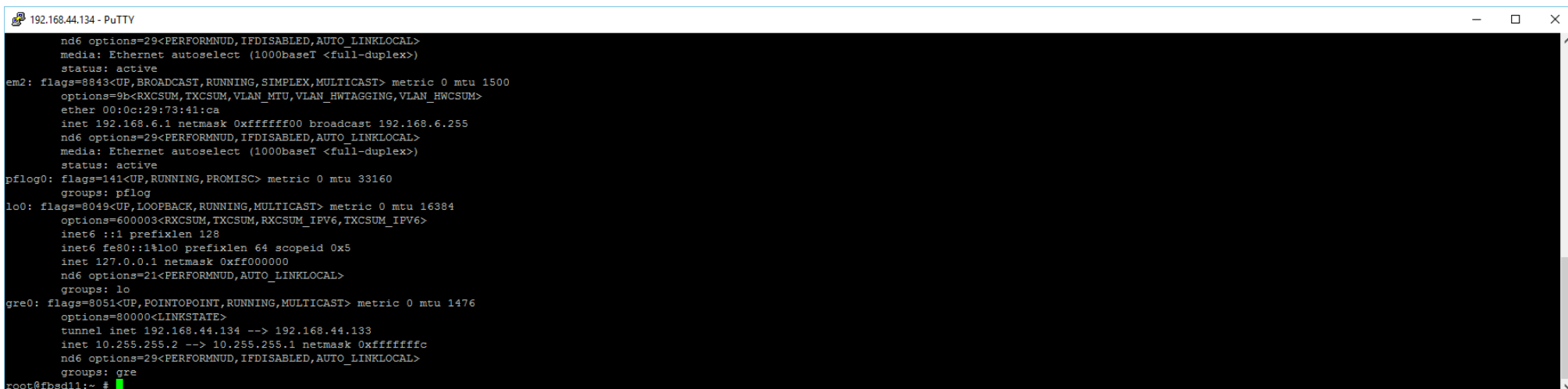
To start strongswan:

$ su

# service strongswan start

IpSEC configuration with Strongswan
```

The proposed configurations make use of strongSwan version 5.4.0 Transversal with NAT functionality already active. For completeness, we will show the main configuration files of the two gateway firewall server.

A screenshot of a terminal window titled "192.168.44.134 - PuTTY". The terminal displays network configuration details for several interfaces. The 'gre0' interface is highlighted, showing it is a tunnel interface with flags 8051 (UP, POINTOPOINT, RUNNING, MULTICAST), metric 0, and mtu 1476. It is configured with a tunnel to 192.168.44.134 and an IP address of 10.255.255.2 with a netmask of 0xfffffff. The terminal also shows configurations for 'em2', 'pflog0', and 'lo0' interfaces.

```
nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
em2: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
ether 00:0c:29:73:41:ca
inet 192.168.6.1 netmask 0xfffff00 broadcast 192.168.6.255
nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
pflog0: flags=141<UP,RUNNING,PROMISC> metric 0 mtu 33160
groups: pflog
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
options=600003<RXCSUM,TXCSUM,RXCSUM_IPV6,TXCSUM_IPV6>
inet6 ::1 prefixlen 128
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x5
inet 127.0.0.1 netmask 0xff000000
nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
groups: lo
gre0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> metric 0 mtu 1476
options=80000<LINKSTATE>
tunnel inet 192.168.44.134 --> 192.168.44.133
inet 10.255.255.2 --> 10.255.255.1 netmask 0xfffffff
nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
groups: gre
root@fbsd11:~ #
```

Figure 6. Gre interface configuration for connection “Fbsd11_01-Fbsd11_02”

In particular, one will show the two firewalling scripts, one realized through ipfw2, the native FreeBSD firewall, and the other with pf, the OpenBSD firewall.

```
192.168.44.133 - PuTTY
em1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=9b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM>
ether 00:0c:29:06:45:16
inet 192.168.3.1 netmask 0xffffffff broadcast 192.168.3.255
nd6 options=29<PERFORMNOD,IFDISABLED,AUTO_LINKLOCAL>
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
em2: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=9b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM>
ether 00:0c:29:06:45:20
inet 192.168.4.1 netmask 0xffffffff broadcast 192.168.4.255
nd6 options=29<PERFORMNOD,IFDISABLED,AUTO_LINKLOCAL>
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
pflog0: flags=0<> metric 0 mtu 33160
groups: pflog
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
options=600003<RXCSUM, TXCSUM, RXCSUM_IPV6, TXCSUM_IPV6>
inet6 ::1 prefixlen 128
inet6 fe80::1::1 prefixlen 64 scopeid 0x5
inet 127.0.0.1 netmask 0xff000000
nd6 options=21<PERFORMNOD,AUTO_LINKLOCAL>
groups: lo
gre0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> metric 0 mtu 1476
options=800000<LINKSTATE>
tunnel inet 192.168.44.133 --> 192.168.44.134
inet 10.255.255.1 --> 10.255.255.2 netmask 0xffffffffc
nd6 options=29<PERFORMNOD,IFDISABLED,AUTO_LINKLOCAL>
groups: gre
user@fbsd11:~ %
```

Figure 7. Gre interface configuration for connection “Fbsd11_02-Fbsd11_01”

In particular, one will show the two firewalling scripts, one realized through ipfw2, the native FreeBSD firewall, and the other with pf, the OpenBSD firewall.

```
192.168.44.133 - PuTTY
Edit /etc/motd to change this login announcement.
If you need to ask a question on the FreeBSD-questions mailing list then

http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/\
freebsd-questions/index.html

contains lots of useful advice to help you get the best results.
user@fbsd11:~ % su -
Password:
root@fbsd11:~ # ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, FreeBSD 11.0-CURRENT, amd64):
  uptime: 109 seconds, since Apr 15 21:50:41 2016
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 3
  loaded plugins: charon aes des blowfish rc2 sha2 sha1 md4 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem openssl fips-prf xcbc cmac hmac attr kernel-pfkey kernel-pfroute resolve socket-default stroke vici updown eap-identity eap-md5 eap-mschapv2 eap-tls eap-ttls eap-peap xauth-generic whitelist addrblock
Listening IP addresses:
  192.168.44.133
  192.168.44.35
  192.168.44.36
  192.168.44.37
  192.168.3.1
  192.168.4.1
  10.255.255.1
Connections:
Fbsd11_01-Fbsd11_02: 10.255.255.1...10.255.255.2 IKEv1/2
Fbsd11_01-Fbsd11_02: local: [10.255.255.1] uses pre-shared key authentication
Fbsd11_01-Fbsd11_02: remote: [10.255.255.2] uses pre-shared key authentication
Fbsd11_01-Fbsd11_02: child: 192.168.3.0/24[gre] == 192.168.5.0/24[gre] TRANSPORT
Security Associations (1 up, 0 connecting):
Fbsd11_01-Fbsd11_02[1]: ESTABLISHED 109 seconds ago, 10.255.255.1[10.255.255.1]...10.255.255.2[10.255.255.2]
Fbsd11_01-Fbsd11_02[1]: IKEv2 SPIs: 8063d5b63f97724c_i* fa8741f9f6cbab26_r, pre-shared key reauthentication in 7 hours
Fbsd11_01-Fbsd11_02[1]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1536
Fbsd11_01-Fbsd11_02[1]: INSTALLED, TUNNEL, reqid 1, ESP SPIs: c3e33c1b_i c7bdfd6c_o
Fbsd11_01-Fbsd11_02[1]: AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 41 minutes
Fbsd11_01-Fbsd11_02[1]: 192.168.3.0/24[gre] == 192.168.5.0/24[gre]
root@fbsd11:~ #
```

```
192.168.44.134 - PuTTY
192.168.5.1
192.168.6.1
10.255.255.2
Connections:
Fbsd11_02-Fbsd11_01: 10.255.255.2...10.255.255.1 IKEv1/2
Fbsd11_02-Fbsd11_01: local: [10.255.255.2] uses pre-shared key authentication
Fbsd11_02-Fbsd11_01: remote: [10.255.255.1] uses pre-shared key authentication
Fbsd11_02-Fbsd11_01: child: 192.168.5.0/24[gre] == 192.168.3.0/24[gre] TRANSPORT
Security Associations (0 up, 0 connecting):
  none
root@fbsd11:~ # ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, FreeBSD 11.0-CURRENT, amd64):
  uptime: 2 hours, since Apr 15 19:19:16 2016
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 8
  loaded plugins: charon aes des blowfish rc2 sha2 sha1 md4 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem openssl fips-prf xcbc cmac hmac attr kernel-pfkey kernel-pfroute resolve socket-default stroke vici updown eap-identity eap-md5 eap-mschapv2 eap-tls eap-ttls eap-peap xauth-generic whitelist addrblock
Listening IP addresses:
  192.168.44.134
  192.168.44.38
  192.168.44.39
  192.168.44.40
  192.168.5.1
  192.168.6.1
  10.255.255.2
Connections:
Fbsd11_02-Fbsd11_01: 10.255.255.2...10.255.255.1 IKEv1/2
Fbsd11_02-Fbsd11_01: local: [10.255.255.2] uses pre-shared key authentication
Fbsd11_02-Fbsd11_01: remote: [10.255.255.1] uses pre-shared key authentication
Fbsd11_02-Fbsd11_01: child: 192.168.5.0/24[gre] == 192.168.3.0/24[gre] TRANSPORT
Security Associations (1 up, 0 connecting):
Fbsd11_02-Fbsd11_01[3]: ESTABLISHED 27 seconds ago, 10.255.255.2[10.255.255.2]...10.255.255.1[10.255.255.1]
Fbsd11_02-Fbsd11_01[3]: IKEv2 SPIs: 8063d5b63f97724c_i fa8741f9f6cbab26_r, pre-shared key reauthentication in 7 hours
Fbsd11_02-Fbsd11_01[3]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1536
Fbsd11_02-Fbsd11_01[3]: INSTALLED, TUNNEL, reqid 2, ESP SPIs: c7bdfd6c_i c3e33c1b_o
Fbsd11_02-Fbsd11_01[3]: AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 46 minutes
Fbsd11_02-Fbsd11_01[3]: 192.168.5.0/24[gre] == 192.168.3.0/24[gre]
root@fbsd11:~ #
```

Figure 8. Ipsec over Gre tunnel is up

On the Fbsd11 Gw1:

Listing 1. The /etc/rc.conf file

```
hostname="fbsd11"

ifconfig_em0="inet 192.168.44.133 netmask 0xffffffff0"

ifconfig_em0_alias0="inet 192.168.44.35 netmask 255.255.255.255"

ifconfig_em0_alias1="inet 192.168.44.36 netmask 255.255.255.255"
```



```
ifconfig_em0_alias2="inet 192.168.44.37 netmask 255.255.255.255"

defaultrouter="192.168.44.2"

ifconfig_em1="inet 192.168.3.1 netmask 255.255.255.0"

ifconfig_em2="inet 192.168.4.1 netmask 255.255.255.0"

sshd_enable="YES"

moused_enable="YES"

ntpd_enable="YES"

dumpdev="AUTO"


# DNS / DHCP Section

local_unbound_enable="YES"

dhcpd_enable="YES"                                # dhcpd enabled?

dhcpd_flags="-q"                                  # command option(s)

dhcpd_conf="/usr/local/etc/dhcpd.conf"            # configuration file

dhcpd_ifaces="em1 em2"                            # ethernet interface(s)

dhcpd_withumask="022"                              # file creation mask


# Gateway Firewall setup

gateway_enable="YES"

## ipfw/natd

natd_enable="YES"

natd_interface="em0"

natd_flags="-f /usr/local/etc/natd.conf"
```

```
firewall_enable="YES"

firewall_script="/usr/local/etc/ipfw.rules"

firewall_logging="YES"


# MPD5

mpd_enable="YES"

#mpd_flags="-b -s mpd"

arpproxy_all="YES"

ng_ipfw_load="YES"

ng_nat_load="YES"


# IPSEC BLOCK

strongswan_enable="YES"


# GRE BLOCK

cloned_interfaces="gre0"

ifconfig_gre0="inet 10.255.255.1 10.255.255.2 netmask 255.255.255.252
tunnel 192.168.44.133 192.168.44.134"

static_routes="tunnel"

route_tunnel="192.168.5.1/24 10.255.255.2"
```

Listing 2. The /boot/loader.conf file

```
wlan_wep_load="YES"

wlan_tkip_load="YES"
```

```
wlan_ccmp_load="YES"

wlan_xauth_load="YES"

wlan_acl_load="YES"

#nvidia_load="YES"

if_gre_load="YES"

ng_car_load="YES"

if_bridge_load="YES"

if_tap_load="YES"

aio_load="YES"

carp_load="YES"

pf_load="YES"

pflog_load="YES"

aesni_load="YES"           #IPSEC

ipfw_load="YES"

ipfw_nat_load="YES"

ipdivert_load="YES"

net.inet.ip.fw.default_to_accept="1"

net.inet.ip.forwarding=1

libalias_load="YES"

ng_mppc_load="YES"

ng_one2many_load="YES"

ng_ppp_load="YES"

ng_pppoe_load="YES"
```

```
ng_pptpgre_load="YES"
ng_rfc1490_load="YES"
ng_socket_load="YES"
ng_bridge_load="YES"
ng_cisco_load="YES"
ng_echo_load="YES"
ng_ether_load="YES"
```

Listing 3. The `/usr/local/etc/ipsec.conf` file

```
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup

    # strictcrlpolicy=yes

    # uniqueids = no

# Add connections here.

conn Fbsd11_01-Fbsd11_02

    # peer IPs

left=10.255.255.1

    leftid=10.255.255.1

    leftsubnet=192.168.3.0/24

    leftfirewall=yes
```



```
right=10.255.255.2

    rightid=10.255.255.2

    rightsubnet=192.168.5.0/24

    # phase 1 parameters

    ike=aes128-sha1-modp1536!

    ikelifetime=28800s

    # authentication

    authby=secret

    # phase 2 parameters

    esp=aes128-sha1-modp1536!

    lifetime=3600s

    type=transport

    leftprotoport=gre

    rightprotoport=gre

    # startup

    auto=start

    keyingtries=%forever
```

Listing 4. The `/usr/local/etc/ipfw.rules` file

```
#!/bin/sh

# Firewall rules
```

```
# Macros

fwcmd=/sbin/ipfw

ext_if="em0"

int_if0="em1"

int_if1="em3"

dmz_if="em2"

#proxy_ip="10.0.0.1"

high_ports="1024-65535"


...

# Vpn LAN Definitions

lan_net0="192.168.3.0/24"

lan_net1="192.168.5.0/24"

dmz_net="192.168.4.0/24"

#ovpn_lan="10.10.1.0/24"


#Flushing firewall Rules ...

#$fwcmd -f flush

ipfw -q -f flush


#Allow SSH

$fwcmd add 0110 allow tcp from any to me 22 via $ext_if setup
```

```
# Allow Mpd

$fwcmd add 0554 allow tcp from any to any dst-port 1723

$fwcmd add 0555 allow tcp from any 1723 to any

$fwcmd add 0556 allow gre from any to any


# Allow L2tp

$fwcmd add 0557 allow udp from any to me 1701 keep-state

$fwcmd add 0558 allow udp from me to any 1701 keep-state


# Allow Ipsec Vpn

$fwcmd add 0559 allow log esp from any to any

$fwcmd add 0560 allow log ah from any to any

$fwcmd add 0561 allow log ipencap from any to any

$fwcmd add 0562 allow log udp from any 500 to any


# VPN Block


# MPD Vpn Rulesets

$fwcmd add 0671 allow gre from any to any

$fwcmd add 0672 allow all from any to any via ng0

...

$fwcmd add 0680 allow all from any to any via ng8
```

```
###  
  
#   NAT Section Start  
  
###  
  
...  
  
### Matching Internal/external IPs for Natting ...  
  
###  
  
#   NAT Section End  
  
###  
  
...  
  
#Final Rulesets  
  
  
$fwcmd add 65526 deny log tcp from any to any in  
  
...  
  
$fwcmd add 65534 deny log all from any to any in
```

Listing 5. The `/usr/local/etc/ipsec.secret` file

```
10.255.255.2 : PSK "Ciao123"
```

And we will explain all the configuration sets:

On the Fbsd11 Gw2:

Listing 6. The `/etc/rc.conf` file

```
hostname="fbsd11"  
  
ifconfig_em0="inet 192.168.44.134 netmask 0xffffffff00"
```



```
ifconfig_em0_alias0="inet 192.168.44.38 netmask 255.255.255.255"
ifconfig_em0_alias1="inet 192.168.44.39 netmask 255.255.255.255"
ifconfig_em0_alias2="inet 192.168.44.40 netmask 255.255.255.255"
#ifconfig_em0_alias3="inet 10.10.20.1 netmask 255.255.255.0 group
lanif"

defaultrouter="192.168.44.2"

ifconfig_em1="inet 192.168.5.1 netmask 255.255.255.0"
ifconfig_em2="inet 192.168.6.1 netmask 255.255.255.0"

sshd_enable="YES"

moused_enable="YES"

ntpd_enable="YES"

dumpdev="AUTO"


# DNS DHCP BLOCK

local_unbound_enable="YES"

dhcpd_enable="YES"

dhcpd_flags="-q"

dhcpd_conf="/usr/local/etc/dhcpd.conf"

dhcpd_ifaces="em1 em2"

dhcpd_withumask="022"

# Gateway Firewall setup

gateway_enable="YES"
```

```
# MPD5

mpd_enable="YES"

#mpd_flags="-b -s mpd"

arpproxy_all="YES"


# IPSEC BLOCK

strongswan_enable="YES"


# - Enabling the OpenBSD Firewall

pf_enable="YES"

pf_rules="/usr/local/etc/pf.conf"

pf_flags=""

pflog_enable="YES"

pflog_logfile="/var/log/pf.log"

pflog_flags=""


# Gre Block

cloned_interfaces="gre0"

ifconfig_gre0="inet 10.255.255.2 10.255.255.1 netmask 255.255.255.252
tunnel 192.168.44.134 192.168.44.133"

static_routes="tunnel"

route_tunnel="192.168.3.1/24 10.255.255.1"
```

Listing 7. The /boot/loader.conf file

```
wlan_wep_load="YES"

wlan_tkip_load="YES"

wlan_ccmp_load="YES"

wlan_xauth_load="YES"

wlan_acl_load="YES"

#nvidia_load="YES"

if_gre_load="YES"

ng_car_load="YES"

if_bridge_load="YES"

if_tap_load="YES"

aio_load="YES"

#kqemu_load="YES"

carp_load="YES"

# Load PF kernel modules

pf_load="YES"

pflog_load="YES"

aesni_load="YES"           #IPSEC

ipfw_load="YES"

ipfw_nat_load="YES"

ipdivert_load="YES"

net.inet.ip.fw.default_to_accept="1"

net.inet.ip.forwarding=1

libalias_load="YES"
```

Listing 8. The /usr/local/etc/ipsec.conf file

```
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup
    # strictcrlpolicy=yes
    # uniqueids = no
    # keyexchange=ikev1
    # dpdaction=restart

# Add connections here.

conn Fbsd11_02-Fbsd11_01
    # peer IPs
    left=10.255.255.2
    leftid=10.255.255.2
    leftsubnet=192.168.5.0/24
    leftfirewall=yes
    right=10.255.255.1
    rightid=10.255.255.1
    rightsubnet=192.168.3.0/24
```

```
# phase 1 parameters

ike=aes128-sha1-modp1536!

ikelifetime=28800s

# authentication

authby=secret

# phase 2 parameters

esp=aes128-sha1-modp1536!

lifetime=3600s

type=transport

leftprotoport=gre

rightprotoport=gre

# startup

auto=start

keyingtries=%forever
```

Listing 4. The `/usr/local/etc/ipsec.secret` file

```
10.255.255.2 : PSK "Ciao123"
```

Listing 5. The `/usr/local/etc/ipfw.rules` file

```
###pf.conf FreeBSD 11

lan_net = "{192.168.5.0/24}"

ovpn_net = "{192.168.58.0/24}"

vlan_nets = "{172.16.1.0/24 172.16.2.0/24 172.16.3.0/24}"
```



```
int_if = "em1"
ext_if = "em0"

ovpn_if = " { tun0 tun1 tun2 tun3 tun4 tun5 tun6 tun7 tun8 tun9} "
pptp_if = " { ng0 ng1 ng2 ng3 ng4 ng5 ng6 ng7 ng8 ng9} "
ext_gw = "192.168.44.134"

pptp_ports="{ 47 1723}"
l2tp_ports="{ 1701 1194 }"
ipsec_ports="{ 50 51 500 4500 }"

...

rdr on $ext_if inet proto tcp from any to ($ext_if:0) port
$pptp_ports -> 192.168.5.1

rdr on $ext_if inet proto udp from any to ($ext_if:0) port
$ipsec_ports -> 192.168.5.1

rdr on $ext_if inet proto udp from any to ($ext_if) port $l2tp_ports
-> 192.168.5.1

...

# VPN IPSEC Rule

pass in quick on $ext_if:0 inet proto udp from any to ($ext_if) port
${ipsec_ports} keep state

pass in quick on $ext_if:0 inet proto esp keep state
```

```
192.168.44.134 - PuTTY
directory, or can be installed later with: pkg install en-freebsd-doc
For other languages, replace "en" with a language code like de or fr.

Show the version of FreeBSD installed: freebsd-version ; uname -a
Please include that output and any error messages when posting questions.
Introduction to manual pages: man man
FreeBSD directory layout:      man hier

Edit /etc/motd to change this login announcement.
Time to change your password? Type "passwd" and follow the prompts.
-- Dru <genesis@istar.ca>
user@fbbsd11:~ % su -
Password:
root@fbbsd11:~ # ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, FreeBSD 11.0-CURRENT, amd64):
  uptime: 2 hours, since Apr 15 19:19:16 2016
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled:
    5
  loaded plugins: charon aes des blowfish rc2 sha2 shal md4 md5 random nonce x50
    9 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem o
  penssl fips-prf xcbc cmac hmac attr kernel-pfkey kernel-pfroute resolve socket-d
  efault stroke vici updown eap-identity eap-md5 eap-mschapv2 eap-tls eap-ttls eap
  -peap xauth-generic whitelist addrblock
Listening IP addresses:
  192.168.44.134
  192.168.44.38
  192.168.44.39
  192.168.44.40
  192.168.5.1
  192.168.6.1
  10.255.255.2
Connections:
Fbsd11_02-Fbsd11_01: 10.255.255.2...10.255.255.1 IKEv1/2
Fbsd11_02-Fbsd11_01: local: [10.255.255.2] uses pre-shared key authentication
Fbsd11_02-Fbsd11_01: remote: [10.255.255.1] uses pre-shared key authentication
Fbsd11_02-Fbsd11_01: child: 192.168.5.0/24[gre] == 192.168.3.0/24[gre] TRANS
PORT
Security Associations (0 up, 0 connecting):
  none
root@fbbsd11:~ #
```

Figure 9. Ipsec over Gre tunnel is down

In particular, it shows the shutdown sequence of one of the two end points:

```
192.168.44.134 - PuTTY
root@fbbsd11:~ # tcpdump -nni gre0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on gre0, link-type NULL (BSD loopback), capture size 262144 bytes
19:33:39.346520 IP 10.255.255.2 > 192.168.3.1: ICMP echo request, id 516, seq 0, length 64
19:33:39.346530 IP 192.168.3.1 > 10.255.255.2: ICMP echo reply, id 516, seq 0, length 64
19:33:40.362685 IP 10.255.255.2 > 192.168.3.1: ICMP echo request, id 516, seq 1, length 64
19:33:40.362713 IP 192.168.3.1 > 10.255.255.2: ICMP echo reply, id 516, seq 1, length 64
19:33:41.437784 IP 10.255.255.2 > 192.168.3.1: ICMP echo request, id 516, seq 2, length 64
19:33:41.437811 IP 192.168.3.1 > 10.255.255.2: ICMP echo reply, id 516, seq 2, length 64
19:33:42.513155 IP 10.255.255.2 > 192.168.3.1: ICMP echo request, id 516, seq 3, length 64
19:33:42.513183 IP 192.168.3.1 > 10.255.255.2: ICMP echo reply, id 516, seq 3, length 64
19:33:43.526519 IP 10.255.255.2 > 192.168.3.1: ICMP echo request, id 516, seq 4, length 64
19:33:43.526547 IP 192.168.3.1 > 10.255.255.2: ICMP echo reply, id 516, seq 4, length 64
19:33:44.539435 IP 10.255.255.2 > 192.168.3.1: ICMP echo request, id 516, seq 5, length 64
19:33:44.539462 IP 192.168.3.1 > 10.255.255.2: ICMP echo reply, id 516, seq 5, length 64
^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
root@fbbsd11:~ # tcpdump -nni em2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on em2, link-type EN10MB (Ethernet), capture size 262144 bytes
^C

192.168.44.134 - PuTTY
root@fbbsd11:~ # ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: icmp_seq=0 ttl=64 time=0.240 ms
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=0.805 ms
64 bytes from 192.168.3.1: icmp_seq=2 ttl=64 time=0.917 ms
64 bytes from 192.168.3.1: icmp_seq=3 ttl=64 time=0.771 ms
64 bytes from 192.168.3.1: icmp_seq=4 ttl=64 time=0.760 ms
^C
-- 192.168.3.1 ping statistics --
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.240/0.699/0.917/0.236 ms
root@fbbsd11:~ # tail -f /var/
account/ at/ audit/ authpf/ backups/ cache/ crash/ cron/ db/ empty/ games/ heimdal/ log/ mail/
root@fbbsd11:~ # tail -f /var/log/
auth.log cron devd.log ipfw.log maillog messages.0.bz2 mpd.log ppp.log sendmail
badinstall.log debug.log dhcpld.log lpd-errors messages messages.1.bz2 pf.log security sendmail
root@fbbsd11:~ # tail -f /var/log/
auth.log cron devd.log ipfw.log maillog messages.0.bz2 mpd.log ppp.log sendmail
badinstall.log debug.log dhcpld.log lpd-errors messages messages.1.bz2 pf.log security sendmail
root@fbbsd11:~ # tail -f /var/log/auth.log
Apr 15 19:19:18 fbbsd11 sshd[878]: Server listening on 0.0.0.0 port 22.
Apr 15 19:19:24 fbbsd11 charon: 14[IKE] 10.255.255.1 is initiating an IKE_SA
Apr 15 19:19:24 fbbsd11 charon: 08[IKE] IKE_SA Fbsd11_02-Fbsd11_01[2] established between 10.255.255.2[10.255.255.2]...10.255.255.1[10.255.255.1]
Apr 15 19:19:24 fbbsd11 charon: 08[IKE] CHILD SA Fbsd11_02-Fbsd11_01[2] established with SPIs c5bfff8_1 c32d0de1_0 and TS 192.168.5.0/24
Apr 15 19:19:28 fbbsd11 login: login on ttyv0 as root
Apr 15 19:19:28 fbbsd11 login: ROOT LOGIN (root) ON ttyv0
Apr 15 19:19:34 fbbsd11 charon: 08[IKE] deleting IKE_SA Fbsd11_02-Fbsd11_01[1] between 10.255.255.2[10.255.255.2]...10.255.255.1[10.255.255.1]
Apr 15 19:19:34 fbbsd11 charon: 14[IKE] IKE_SA deleted
Apr 15 19:20:18 fbbsd11 sshd[965]: Accepted keyboard-interactive/pam for user from 192.168.44.1 port 11238 ssh2
Apr 15 19:20:44 fbbsd11 su: user to root on /dev/pts/0
```

Figure 10. Ipsec over Gre tunnel logs

Here, one use's tcpdump to capture output over gre:

```
# tcpdump -nni gre0
```

Conclusions

GRE over IPSEC technology makes it possible to create VPN between different geographic locations, it is available on all versions of BSD and Linux, and can be integrated into the kernel by adding the opportune options. It also offers a level of security and very high flexibility, allowing more end-to-end contemporary connections. It is therefore an excellent alternative to using dedicated hardware, expensive equipment provided by industry leading vendors such as Cisco Systems, with which it is able to communicate by means of the correct configuration; additionally, a plus is that it is open source. The decision to implement gateway portal gre over ipsec, especially in small business environments simplifies the work of system administrators; thus, allowing them to get good performance, saving time, money and effort.

About the Author:

Antonio Francesco Gentile lives in Italy, Calabria, is a software and network engineer. He works for CNR, the National research center as network manager, with the “Culture Lab” <http://culture.deis.unical.it> Department of Telematics at University of Calabria, the computer science associations “Hacklab Cosenza” <http://hacklab.cosenzainrete.it/> and “Verde Binario” <http://www.verdebinario.org/> and is a freelance columnist for Italian magazines “Linux&C” <http://www.oltrelinux.com/> , “Linux Magazine” <http://www.linux-magazine.it/> and “Elettronica OpenSource” <http://it.emcelettronica.com/>.

Reusing the OpenBSD arc4random in Multithreaded User Space Programs

by Sudhi Herle

Recently, a friend asked me for some help in speeding up crypto operations in her multi-threaded (pthreads) Linux program. She was using the standard `/dev/urandom` interface to generate random bytes. And, profiling showed that reading from `/dev/urandom` took a lot of time. I recalled that OpenBSD folks have a high quality userspace cryptographic random number generator called `arc4random`. But to the best of my knowledge, it was not generally available for use in any Linux program. And so, it seemed like an excellent weekend project to create a portable version of the OpenBSD `arc4random(3)` generator for use in multi-threaded programs.

Before Starting:

1. Make the latest version of OpenBSD `arc4random(3)` portable across Unixes and in particular Linux.
2. No global state and no locks - so multi-threaded programs can work cleanly.
3. No API change for programs; i.e., regardless of whether they were single-threaded or multi-threaded, the API should stay the same.

Making OpenBSD arc4random(3) thread-safe

The starting point was the arc4random(3) in OpenBSD libc. First was use of global variables for storing random state:

```
static struct _rs {
    size_t    rs_have;    /* valid bytes at end of rs_buf */
    size_t    rs_count;   /* bytes till reseed */
} *rs;

/* May be preserved in fork children, if _rs_allocate() decides. */
static struct _rsx {
    chacha_ctx rs_chacha; /* chacha context for random keystream */
    u_char     rs_buf[RSBUFSZ]; /* keystream blocks */
} *rsx;
```

As you can see, “rs” and “rsx” are both global variables.

The rest of the arc4random.c source code refers to these global state variables; so, the first task was to change the functions to take an additional “state” argument. I collected all the state information needed into one struct:

```
/* ChaCha20 state */
typedef struct
{
    uint32_t input[16]; /* could be compressed */
} chacha_ctx;
```



```
/* arc4random state. */

struct rand_state
{
    size_t      rs_have;      /* valid bytes at end of rs_buf */
    size_t      rs_count;     /* bytes till reseed */
    pid_t       rs_pid;       /* My PID */
    chacha_ctx  rs_chacha;    /* chacha context for random
keystream */
    u_char      rs_buf[ARC4R_RSBUFSZ]; /* keystream blocks */
};

typedef struct rand_state rand_state;
```

The ‘rand_struct’ was used as the “this” pointer in every function that originally operated on the global variables. For example, `_rs_init()` went from this:

```
/* Original OpenBSD version of _rs_init() */

static inline void
_rs_init(u_char *buf, size_t n)
{
    if (n < KEYSZ + IVSZ)
        return;

    if (rs == NULL) {
        if (_rs_allocate(&rs, &rsx) == -1)
            abort();
    }
}
```

```
chacha_keysetup(&rsx->rs_chacha, buf, KEYSZ * 8, 0);
    chacha_ivsetup(&rsx->rs_chacha, buf + KEYSZ);
}
```

To this:

```
/* Revised version of _rs_init() that takes a context arg */
static inline void
_rs_init(rand_state* st, u8 *buf, size_t n)
{
    assert(n >= (ARC4R_KEYSZ + ARC4R_IVSZ));

    chacha_keysetup(&st->rs_chacha, buf, ARC4R_KEYSZ * 8, 0);
    chacha_ivsetup(&st->rs_chacha, buf + ARC4R_KEYSZ);
}
```

The change above was replicated to each of the internal functions. The modified version looks like so:

```
void      _rs_init(rand_state* st, u8* buf, size_t n)
void      _rs_rekey(rand_state* st, u8* buf, size_t n)
void      _rs_stir(rand_state* st)
void      _rs_stir_if_needed(rand_state* st, size_t n)
void      _rs_random_buf(rand_state* st, void* buf, size_t n)
uint32_t  _rs_random_u32(rand_state*)
```

That leaves us with the external facing functions - `arc4random_buf()` and `arc4random()` and `arc4random_uniform()`. Each of these have a stable API used by userspace programs. The original functions used locks to protect the global variables. For reference, this is the original version from OpenBSD:

```
uint32_t
arc4random(void)
{
    uint32_t val;

    _ARC4_LOCK();

    _rs_random_u32(&val);

    _ARC4_UNLOCK();

    return val;
}
```

We will be doing some clever things with pthreads to remove the locks. Before we do that, it's useful to recap some functions in the POSIX pthreads API that will aid us in our quest.

Brief Interlude - POSIX Threads

On a modern system that supports POSIX threads API (pthreads API), programs are allowed to create special slots to stash per-thread data. This is accomplished by calling the `pthread_getspecific()` API. Each piece of data needs to have a unique “key” associated with it. The per-thread data is set using the `pthread_setspecific()` API. However, before either of these APIs can be called, the key must first be created by calling `pthread_key_create()`. It is most critical that this key-creation is done only ONCE per key, per process.

In order to guarantee that certain functions are only ever called once per process, pthread provides a convenience function called `pthread_once()`. This function takes a function pointer as an argument and guarantees that that function is called exactly once.

Multi-thread safety

Now, we have assembled most of the pieces to complete the puzzle:

- We have removed global variables and added a new context variable for each of the arc4random internal functions
- We know about the pthreads `get_specific()` and `set_specific()` APIs to manage per-thread context

We are now ready to make `arc4random()` thread-aware and thread-safe. First, we create the function that runs exactly once per process and creates the necessary key for `pthread_getspecific()` and `pthread_setspecific()`:

```
/* Global variables used for pthread_once() and random-state key */
static pthread_once_t Ronce = PTHREAD_ONCE_INIT;
static pthread_key_t  Rkey;
static Uint32_t       Rforked = 0;

static void screate()
{
    pthread_key_create(&Rkey, 0);
    pthread_atfork(0, 0, atfork);
}
```

We will return to `pthread_atfork()` later. Now, `arc4random()` becomes very simple:

```
Uint32_t arc4random()
{
    rand_state* z = sget();
    return _rs_random_u32(z);
}
```

The function “sget()” does a few things::

- Creates the key needed for pthread_get_specific()
- Allocates state for the calling thread if needed and initializes random state if needed
- Finally, returns the per thread state

Its implementation is quite simple:

```
static rand_state* sget()
{
    pthread_once(&Ronce, screate);

    volatile pthread_key_t* k = &Rkey;
    rand_state * z = (rand_state *)pthread_getspecific(*k);
    if (!z) {
        z = (rand_state*)calloc(sizeof *z, 1);
        assert(z);

        _rs_stir(z);
        z->rs_pid = getpid();

        pthread_setspecific(*k, z);
    }

    << fork check >>

    return z;
}
```


The first thing it does is setup for “screate()” to be called exactly once per process. When `pthread_once()` returns, it is guaranteed that `screate()` is called exactly once. So, when the call to `pthread_getspecific()` is made, either the per-thread random context is available or not. If this is the first time that `sget()` is called by this thread, then obviously, no per-thread context is available yet. So, it allocates memory, calls the required initialization code and finally, it sets the per-thread state by calling `pthread_setspecific()`.

With the above changes, we have accomplished the following:

- Eliminated the use of global variables for storing the random state; therefore, no need for locks to protect global variables
- Created a per-thread state for the random generator - so each thread gets its own generator instance

Fork Safety

In order to ensure that the random generator state is not shared when a process forks, we need to detect when calls to `fork()` are made. Therefore, when we obtain the per-thread state, we check to see if the process has forked. We use two ways to detect this:

1. An atomic counter that is incremented every time the process forks
2. Current process pid compared against the pid in the random generator state

Recall the `screate()` function above - where we saw a call to `pthread_atfork()`. This pthreads function arranges for the supplied function to be called in the child process after `fork()`. Our implementation of this callback function is quite simple. We increment a variable atomically.

In the `sget()` function, we use this atomic variable and the current pid to detect if a fork has happened:

```
<< fork check >>

if (Rforked > 0 || getpid() != z->rs_pid) {
    __sync_fetch_and_sub(&Rforked, 1);
    z->rs_pid = getpid();
}
```

```
    _rs_stir(z);  
}
```

In case the fork did happen, we reset the generator state and update the pid.

Making it Portable

The last remaining bit is to make a portable version of OpenBSD `getentropy()` call. This I accomplished by making it a generic function that is implemented differently by each platform. On most Unix like platforms, that function is as simple keeping the file descriptor open and reading from `/dev/random` as needed. In the github repository below, a sample implementation of `getentropy()` for Linux is provided in the file `posix_entropy.c`.

Results

The source code for the portable generator and a simple benchmark is in github: <https://github.com/opencoff/mt-arc4random>

To compile the example benchmark:

```
git clone https://github.com/opencoff/mt-arc4random.git  
  
cd mt-arc4random  
  
make
```

The benchmark program is called “`t_arc4rand`”. It displays CPU cycles/per byte for each size. The “`sysrand`” column indicates the speed of reading from `/dev/urandom`.

When run on a retina MacBook Pro 13” (2013) running OS X Yosemite:

size,	arc4rand,	sysrand,	speed-up
16,	12.2966,	279.9628,	22.77
32,	11.3687,	268.2029,	23.59
64,	9.9161,	238.8743,	24.09
256,	9.3164,	217.1194,	23.30
512,	8.3054,	204.1270,	24.58

As you can see, on OS X, the arc4random generator we just built runs almost 20x faster compared to reading from /dev/urandom!

On Debian Linux (sid) x86_64 on a Core-i7 laptop running the Linux 4.5 kernel, I see:

size,	arc4rand,	sysrand,	speed-up
16,	9.6997,	255.6211,	26.35
32,	7.9821,	251.2072,	31.47
64,	7.5916,	220.8430,	29.09
256,	7.4764,	206.5079,	27.62
512,	7.2225,	201.9913,	27.97

That's a nice speedup compared to reading from /dev/urandom.

The generator consumes approx 1.5kB of state per thread.

Closing Notes

The userspace port of the OpenBSD arc4random generator is useful in many projects that need a very high speed cryptographic quality random source. Its high performance is particularly beneficial to embedded systems.

As a matter of good programming discipline, the design pattern outlined in the article is useful for anyone working with multi-threaded programs. If you think of having your internal library functions work on global variables, don't do that.

Instead, pass the necessary information as “context” in the first argument of the functions. At some point, if you really must use a global variable, then the technique above allows you to create per-thread state in a lockless manner.

Source Code

The source code for this is available in github: <https://github.com/opencoff/mt-arc4random>

About the Author

Sudhi Herle runs a large engineering and product management organization at RhythmOne - an ad-tech company. In his free time, he works on interesting programs. His professional page is on LinkedIn (<https://www.linkedin.com/in/sudhiherle>) sudhi@herle.net

HOW TO install the XFCE 4.12 Desktop on NetBSD 7

by Curt McIntosh

Before Starting:

<https://slice2.com/2016/01/30/howto-install-the-xfce-4-12-desktop-on-netbsd-7/>

This is an update to previous posts for NetBSD 6x:

<http://slice2.com/2015/01/03/howto-install-the-xfce-4-desktop-on-netbsd-6-1-5/>

<http://slice2.com/2013/10/10/howto-install-the-xfce-4-desktop-on-netbsd-6-1-2/>

For a lightweight functional desktop on NetBSD, install XFCE. As root, perform the following steps. This covers 32 and 64 bit x86 hardware. Since NetBSD essentially runs on everything, simply adjust the repository path to your architecture from the list here: <http://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/>

Setup your binary repository

```
mkdir -p /usr/pkg/etc/pkgin
touch /usr/pkg/etc/pkgin/repositories.conf
vi /usr/pkg/etc/pkgin/repositories.conf
```

and add path:

For x64

http://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/amd64/7.0_2016Q1/All/

For x32

http://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/i386/7.0_2016Q1/All/

Add the NetBSD ftp server to your host file

This is for convenience and can be removed when done.

```
vi /etc/hosts and add:  
  
199.233.217.201 ftp.netbsd.org
```

Export your path

Note: I do not know why the encoded quote characters keep appearing after /ALL/ in the path statements below. It must be an html coding issue and I am not a developer. Just make sure that at the end of the path statement it ends with /7.0_2016Q1/ALL/" with no trailing characters. In other words, it should look like the paths depicted in step 1 above, only it must end in a " character.

For x64:

```
export  
PKG_PATH="http://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/amd64/7.0_2016Q1/All/"
```

For x32:

```
export  
PKG_PATH="http://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/i386/7.0_2016Q1/All/"
```

Install the latest version of pkgin on your system

```
pkg_add -v pkgin-*
```

Update the pkgin database and install XFCE

```
pkgin update  
  
pkgin install xfce
```

calculating dependencies... done. Nothing to upgrade.

121 packages to be installed (251M to download, 887M to install):

```
nettle-3.1.1 libtasn1-4.5 libcfg+-0.6.2nb3 gmp-6.0.0a libproxy-0.4.11
libgpg-error-1.20 libcddeb-1.3.2nb1
p5-Business-ISBN-Data-20140910.002nb1 py27-cElementTree-2.7.10
libIDL-0.8.14nb4 at-spi2-core-2.16.0 icu-55.1nb1 libepoxy-1.3.1nb1
at-spi2-atk-2.16.0 ORBit2-2.14.19nb4 gobject-introspection-1.44.0
p5-Business-ISBN-2.09nb1 usbids-20081118 pciids-20150907
libvolume_id-0.81.1nb1 hal-info-20091130nb4 libcdio-0.93nb3
libgcrypt-1.6.4 glib-networking-2.36.2nb2 readline-6.3nb3
popt-1.16nb1 mit-krb5-1.10.7nb7 libiconv-1.14nb2 gnutls-3.3.18
gettext-lib-0.19.4 jbigkit-2.1 fribidi-0.19.7 enca-1.15 libogg-1.3.2
libidn-1.32 xvidcore-1.3.3 x264-devel-20150717 libvpx-1.4.0nb1
libtheora-1.1.1nb2 libass-0.12.2 lame-3.99.5nb3 tiff-4.0.6 lcms2-2.7
poppler-0.34.0 samba-3.6.25nb2 libsoup-2.50.0 libgnome-keyring-3.12.0
libcdio-paranoia-0.93nb1 hal-0.5.14nb16 p5-URI-1.69 xcb-util-0.4.0
libvorbis-1.3.5 libltdl-2.4.2 gstreamer0.10-0.10.36nb8
GConf-2.32.4nb10 iso-codes-3.61 gtk3+-3.16.6nb1 xmlcatmgr-2.2nb1
perl-5.22.0 pcre-8.38 libelf-0.8.13nb1 lzo-2.09 harfbuzz-1.0.3
cairo-gobject-1.14.2nb1 libffi-3.2.1 libxml2-2.9.2nb3
gnome-icon-theme-3.12.0 shared-mime-info-1.4 python27-2.7.10
py27-expat-2.7.10 pango-1.37.1 cairo-1.14.2nb1 atk-2.16.0
gtksourceview2-2.10.5nb24 glib2-2.44.1nb1 policykit-0.9nb18
xfce4-garcon-0.5.0 xfce4-conf-4.12.0nb2 libxklavier-5.0nb5
libglade-2.6.4nb22 libcanberra-0.27nb5 vte-0.28.1nb16
startup-notification-0.12nb3 xfce4-exo-0.10.6 libxfce4util-4.12.1nb1
libnotify-0.7.6nb2 libexif-0.6.21 gvfs-1.6.7nb17 poppler-glib-0.34.0
png-1.6.20 openjpeg-2.1.0 libgsf-1.14.34 jpeg-9anb1
gdk-pixbuf2-2.30.8nb1 ffmpegthumbnailer-2.0.8nb4 ffmpeg1-1.2.12nb1
dbus-glib-0.104 dbus-1.10.0nb1 curl-7.44.0 libxfce4ui-4.12.1nb2
libwnck-2.30.6nb18 hicolor-icon-theme-0.13 desktop-file-utils-0.22
xfce4-xarchiver-0.5.4nb1 xfce4-wm-themes-4.10.0nb1 xfce4-wm-4.12.3
xfce4-tumbler-0.1.31nb3 xfce4-thunar-1.6.10nb1
xfce4-terminal-0.6.3nb1 xfce4-settings-4.12.0nb1 xfce4-session-4.12.1
xfce4-panel-4.12.0nb1 xfce4-orage-4.12.1 xfce4-mousepad-0.4.0nb1
xfce4-gtk2-engine-3.2.0nb1 xfce4-desktop-4.12.3
xfce4-appfinder-4.12.0nb1 gtk2+-2.24.28
elementary-xfce-icon-theme-0.6 xfce4-4.12.0nb2
```

proceed ? [Y/n] **Y**

Add fonts, fam, screen lock and file manager

```
pkgin install font-adobe-75*
pkgin install font-adobe-100*
pkgin install font-adobe-utopia*
pkgin install xscreensaver
pkgin install fam
pkgin install tbd (dependency of thunar)
pkgin install gvfs (dependency of thunar)
pkgin install xfce4-thunar

cp /usr/pkg/share/examples/rc.d/famd /etc/rc.d/
cp /usr/pkg/share/examples/rc.d/dbus /etc/rc.d/
cp /usr/pkg/share/examples/rc.d/hal /etc/rc.d/

echo rpcbind=YES >> /etc/rc.conf
echo famd=YES >> /etc/rc.conf
echo dbus=YES >> /etc/rc.conf
echo hal=YES >> /etc/rc.conf

/etc/rc.d/rpcbind start

/etc/rc.d/famd start

/etc/rc.d/dbus start

/etc/rc.d/hal start
```

Configure X and start the desktop for the first time

Note that you should not start X as root. Run the following for users on the system. For example, the user slice2 would be setup as:

```
echo xfce4-session >> /home/slice2/.xinitrc

ln /home/slice2/.xinitrc /home/slice2/.xsession

su - slice2

startx
```

Note: be patient, it may take a minute to load

When prompted, select use default config. In the upper left, select Applications > Log out.

Install apps as desired

This step is optional. Enter Y when asked to proceed ? [Y/n] for each app.

Browsers and plugins:

```
pkgin install firefox
pkgin install opera
pkgin install xpdf
pkgin install flashplayer
pkgin install openquicktime
pkgin install mozilla-fonts*
pkgin install icedtea-web
```

When done installing icedtea-web, run the three commands below to configure avahi.

```
cp /usr/pkg/share/examples/rc.d/avahidaemon /etc/rc.d/avahidaemon
chmod 0755 /etc/rc.d/avahidaemon
echo avahidaemon=YES >> /etc/rc.conf
```

Install security apps, utils and shells:

```
pkgin install wireshark
pkgin install nmap
pkgin install iftop
pkgin install keepassx
pkgin install bash
pkgin install lsof
pkgin install mhash
```

```
pkgin install nbtscan  
pkgin install netcat  
pkgin install vim
```

GUI ftp/scp client:

```
pkgin install filezilla
```

Office Suite and multimedia:

```
pkgin install libreoffice*  
pkgin install xmms  
pkgin install xfce4-xmms-plugin  
pkgin install xcdroast  
pkgin install xcalc  
pkgin install vlc  
pkgin install tree
```

You can launch libreoffice from Applications > Office, or enter the soffice command in an xterm.

Now that all your apps are installed, start your desktop

```
su - slice2 (su to your user account)  
  
startx (remember, it takes a minute to load)
```


About the Author:

Curt McIntosh is a Senior Infrastructure Engineer with RiptideTechnology. A true jack of all trades, his experience spans multiple operatingsystems, applications and hardware platforms. Holding certifications such as CEH,SSCP, Linux+, Security+, and Cloud+, NCTA, VCP5-DCV, CCNA, MCSE, MCNE, hispresent focus is on securing systems from the OS to the network. To see more, visit the slice2.com blog. It's a blend of odd and obscure configuration steps that are either poorly documented or not documented at all by the vendor.

FreeBSD Flavors. Do We Need Them? Today...GhostBSD

by George Bungarzescu

A (not too deep) journey to GhostBSD - desktop and enterprise options - compared to pure FreeBSD

Years ago, I used Microsoft products, from operating systems to servers at my job. I always felt that, even if they have good products, the strategy to be closed source, even succesful from one point of view, can not cover the entire requirements of information world. Becoming a more skilled user, I have opened the doors to the wonderful world of Linux, BSD and other open source operating systems and products.

Right now, I use at home for everyday tasks a Linux distro. I feel comfortable but I feel that somehow, in order to have a more user friendly environment, there is a continuous need for improvement. I will not go into why using Microsoft products can be the wrong strategy, but I will state that it is required to have alternatives, and, diversity is a good thing for everyone (just remember how some virus, ransomware and other malware have affected business these days).

But wait, we are talking about Linux not FreeBSD, right? No. Actually, as you already know, the line between Linux and BSD is gray as they share a common history and open source ecosystem software.

So let`s talk a little bit about the history of Unix, BSD and Linux (the advanced users can skip the next part).

Berkeley Software Distribution (BSD) is a generic name for operating systems that are close to the original UNIX design. Having in mind "a robust, general purpose, time-sharing computing platform which would not become obsolete every time the hardware change" in 1977 - many years before Linux was born, a team created the first version of a BSD-like system.

According to Wikipedia "FreeBSD's roots go back to the University of California, Berkeley. The university acquired a UNIX source license from AT&T. Students of the university started to modify and improve the AT&T Unix and called this modified version Berkeley Unix or BSD, implementing features such asTCP/IP, virtual memory and the Unix File System.

The BSD project was founded in 1976 by Bill Joy. But since BSD contained code from AT&T Unix, all recipients had to get a license from AT&T first in order to use BSD.

In June 1989, "Networking Release 1" or simply Net-1 – the first public version of BSD – was released. After releasing Net-1, Keith Bostic, a developer of BSD, suggested replacing all AT&T code with freely-redistributable code under the original BSD license. Work on replacing AT&T code began and, after 18 months, much of the AT&T code was replaced. However, six files containing AT&T code remained in the kernel. The BSD developers decided to release the "Networking Release 2" without those six files. Net-2 was released in 1991".

To be more accurate, the FreeBSD Project itself had its genesis in the early part of 1993, partially as an outgrowth of the Unofficial 386BSDPatchkit (source - FreeBSD manual). The first distribution, FreeBSD 1.0 was released in December of 1993. There are many flavors derived from this old software, but four are most popular, FreeBSD, NetBSD, OpenBSD and Dragon Fly BSD, and they are used to build many of what we know as BSD distros. Some people argue that even Darwin (open source version of Mac OS X) shares a large portion of code with FreeBSD. Also, Microsoft Windows was inspired from and used for production purposes BSD operating systems (hosting for example earlier version of Hotmail and Microsoft website, the sockets design, tools, etc.).

The point is that, before having the internet, DOS, CP/M, Microsoft Windows, Linux and graphical interfaces, we had a good, solid, and ahead-of-its-time operating system that had roots in the Unix software, and provided us with a base to build on. For example, "all modern operating systems implement a version of the Berkeley or POSIX socket interface. It became the standard interface for connecting to the Internet. Even the Winsock implementation for MS Windows, developed by unaffiliated developers, closely follows the standard"(Wikipedia). In fact, many other technologies we have today, would never have been created without the existence of BSD-like operating systems and distros.

Having this understanding of BSD in generic terms, anyone can ask - what is the role of FreeBSD in the modern world and of the flavors we have?

From this perspective, we must understand that most of BSD "distro" are server oriented. Even unknown to the public there are many servers, most of them used for hosting environments that are running on FreeBSD, NetBSD or OpenBSD.

Besides the lack of marketing, there are interests in the open source world to keep alive BSD like systems and this interest came from, as a proof of the quality of code, a working group of a well known distribution, Debian. <https://www.debian.org/ports/kFreeBSD-gnu/>

If on the server side things are clear, I personally found it a little bit hard to configure the graphical interface. But here comes the help of desktop oriented distributions, such as GhostBSD.

Here are some thoughts from **Eric Turgeon**, Development Leader at GhostBSD:

[GB]: What was your reason to start GhostBSD project?

[ET]: Before I started using FreeBSD, I was an Ubuntu user curious about real Unix and hacking software. I found Eric S Raymond's paper on How to be a hacker (<http://www.catb.org/~esr/faqs/hacker-howto.html>). In that paper, he mentioned BSD Unix and I was curious about BSD. I did some research and installed FreeBSD and found that it's not really user friendly. I did some more research and I tried out PCBSD. I was a Gnome user and I did like what PCBSD was trying to achieve, but KDE was not my DE. From there, I wanted to start my own FreeBSD Distribution. This is when I started GhostBSD as a Gnome Alternative to PCBSD, which was KDE only at that time.

[GB]: How closely related is GhostBSD to the BSD family of operating systems?

[ET]: GhostBSD is basically FreeBSD with GTK DE (Desktop Environment), like MATE, XFCE, Cinnamon, and Gnome, all pre-configured and ready to use. Also, we are developing a GUI (Graphical User Interface) tool, like Networkmgr, Update Station, Software Station, GBI (Graphical BSD Installer) and many more tools that are not found on FreeBSD. GhostBSD is using FreeBSD ports tree and pkg repository, there is no change to the default kernel.

[GB]: What are, from your - main developer - point of view, the strengths of GhostBSD compared to FreeBSD? How about the enterprise class of products based on GhostBSD - virtualisation, clustering, etc.?

[ET]: FreeBSD focuses more on a server OS and GhostBSD's focus is Desktop, but since GhostBSD's base system is FreeBSD, GhostBSD is capable to be used on a server. GhostBSD is clearly aimed at the home and office environment. GhostBSD is capable of doing day to day tasks, even gaming.

[BG]: Funding such a project is hard. What part of GhostBSD attracts more funding? How about coding volunteers?

[ET]: Yea, funding a project like GhostBSD is not easy task, GhostBSD is strictly funded by donations, Adsense, by some partnerships and sponsors. Sponsors are basically helping us get more donations because we display a banner or logo linked to our sponsor's website that give them cheap advertising. Recently, we started a Patreon campaign, but so far it has not interested anyone. GhostBSD doesn't generate enough money to finance full time development, so GhostBSD development is only in our spare time. Sometimes, some people contribute to documentation or

[BG]: Are there other proposed changes that will be specific to GhostBSD that will contribute to future adoption? What are your future plans with GhostBSD?

[ET]: There are a lot of features that we would like to see in GhostBSD, like integration of Tor by default ready to use, a tool to enable sshd and other things that are not setup by default. Also, I would like to start a GTK DE build for GhostBSD/FreeBSD.

As a conclusion, BSD like systems are proved to be fast, stable, server and desktop ready. The only weakness we see is a not the lack of information or functionality but exactly having enough BSD flavors to help a user fit exactly their needs. Having more help from volunteers and a modern way of funding such project can also accelerate the development of specific functionalities and growth of BSD like systems adoption as day to day systems. GhostBSD especially offers an easy solution for desktop oriented users by providing a nice and easy to use installer and a familiar Gnome environment.

FreeBSD clones? They are fast, stable and ready for any challenge, so please, use them! And spread the word.

George

P.S. Eric can be reached using [ericturgeon @ ghostBSD.org](mailto:ericturgeon@ghostBSD.org) or directly to ghostBSD.org

About the Author:

I am George, an open source enthusiast working as Senior IT Specialist. I have more than 12 years experience using different operating systems and more than 10 years in software development. I am happy to help, in any way, the open source community and to promote open source based solutions.

You can reach me at george.bungarzescu@gmail.com or via linkedin

PRACTICAL PYTHON WORKSHOP

PEDRO ARAÚJO
& RUI SILVA

Module 1 (introduction):

Why python?

- Introduction about python programming language.
- Learning the strengths of the language and what's good with python.
- Learning where to use Python and why.
- Python as an interpreted language
- How to choose correct interpreter, install it, run it.
- Python virtual environments.
- Text editor (kate, gedit, brackets).
- How to create Hello world, from interpreter and with .py script.
- Standards and batteries included
- Standards and PEP8.
- Batteries included (just to show the most useful python core libraries and link to the official documentation).

Module 2 (python basics):

- Python data types and flow control statements
- Ifs, fors, whiles
- Lists (slices), dictionaries (loop over items), sets
- Python internals
- Classes and object instances
- Everything is an object (docs strings, getters, setters, override)
- Exceptions handling
- Practical example
- Use twitter's API to get some data and show it in the console

Module 3 (files):

- Files
- Duck typing.
- Opening and reading from files.
- Csv files and csvreader.
- Practical exercise

Read file with a sentence per line.

- Manipulate and gather metrics on each sen-

tence.

Output a file with metrics on each sentence.

Module 4 (project):

- Practical project
- Get data from external source

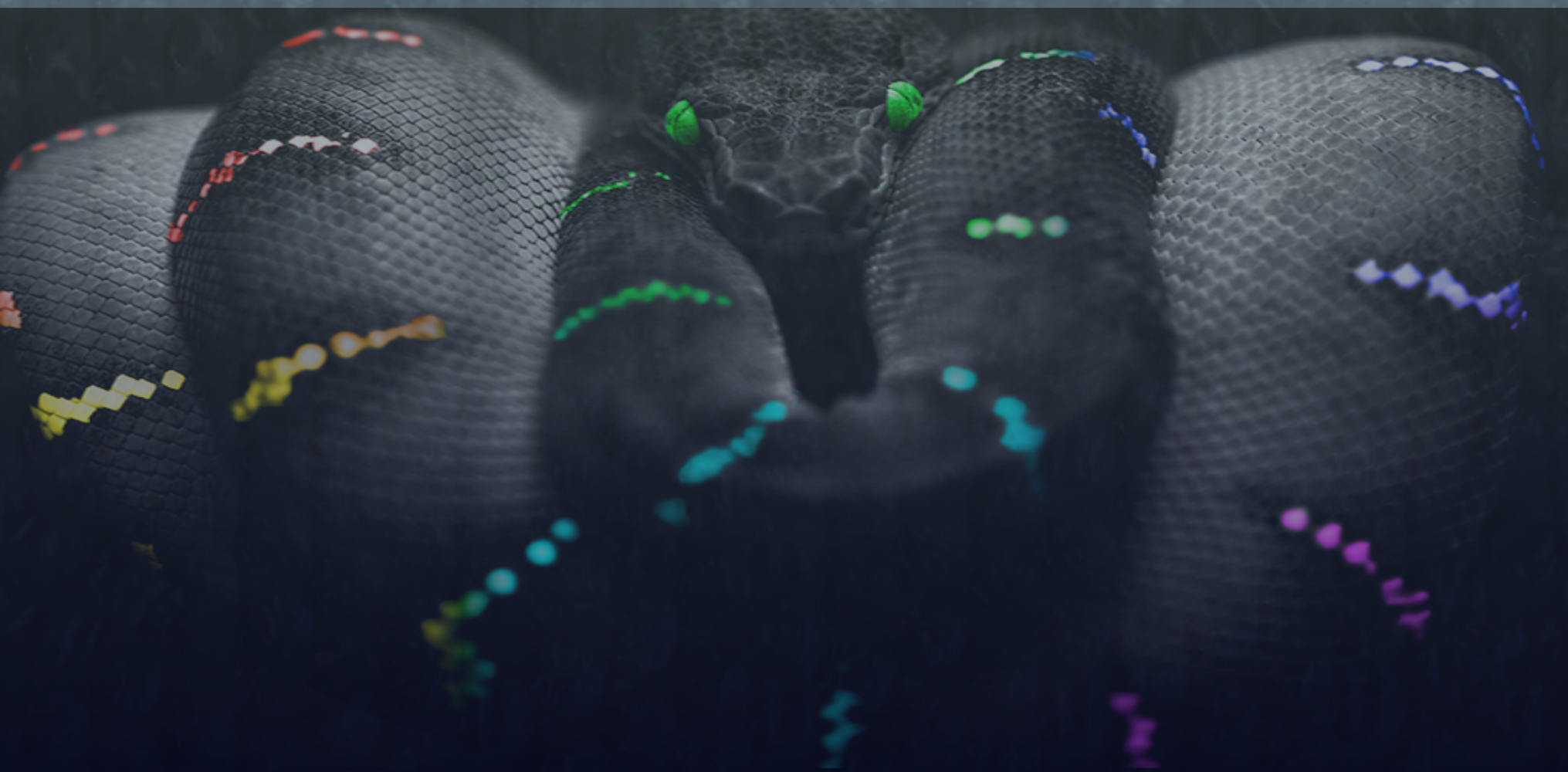
(<http://openweathermap.org>).

- Manipulate data to suit our needs.
- Plot a graph to show the data in a graphical and understandable way.

Authors

Pedro Araujo and Rui Silva

If you have any questions or just want to get to know us better feel free to contact me at marta.ziemianowicz@bsdmag.org or visit <https://bsdmag.org/course/python-programming-coming-next/>



Your ability to put time and effort in a consistent way is the key to success.

Fernando Rodríguez, Co-founder of KeepCoding

by Marta Ziemianowicz, Marta Strzelec & Marta Sienicka

[BSD Magazine]: Hello Fernando, how have you been doing? Can you introduce yourself to our readers?

[Fernando Rodríguez]: My name is Fernando Rodríguez and I'm the co-founder of KeepCoding. I'm an iOS instructor myself and have trained teams around the world, ranging from medium sized companies in Bolivia to Facebook in Menlo Park.

I have recently moved from Madrid, to the San Francisco Bay Area, where we recently created an American subsidiary of our company.

[BSD Mag]: Can you tell us something about KeepCoding?

[FR]: KeepCoding started four years ago in Madrid, Spain, with three students of my first iOS programming course. We now have 5000+ students around the world, mostly Europe and LATAM, with raving reviews. In the Spanish speaking market, we're considered the "Ivy League" (although I prefer the "Justice League" ;-)) of programming bootcamps.

We have recently opened a subsidiary in the Silicon Valley and expect to start operating in the last quarter of this year.

[BSD Mag]: What kind of students can join this school? Should they have any particular experience?

[FR]: Anyone with some programming experience can join. The web bootcamp is certainly more gentle than the Mobile one, but you don't have to be a super talented individual. Quite the opposite, the common factor in our most successful students has always been something else: the willingness to sweat. Your ability to put time and effort in a consistent way is the key to success. Natural talent is a plus, not a must. I've seen many extremely talented developers fail, precisely because they lacked the willingness to sweat.

Most of our students have a full-time job and families. So did Tolstoy. He had 13 children, and yet War and Peace got written. Those who understand that, inevitably succeed.

[BSD Mag]: Where did the idea of such a “school” come from?

[FR]: It came from my own frustration while learning new concepts and tools. When learning a completely new technology, it's very easy to dismiss the forest for paying too much attention to the trees.

The amount of time you can save and the insights you can get when a more experienced developer clearly points out to you what you need to know and why is outstanding. If this mentor is truly a great one, he will even find a way to make the process fun.

This has always been our mission, become a Jedi Academy for programmers to learn nerdy stuff in a fun and efficient way! :-)

[BSD Mag]: And you are a teacher yourself? What do you teach the most? What do you like to teach the most and what would you like to start teaching?

[FR]: Yes, I am a teacher and I believe that's a key part of our success. The owner of a Brazilian airline called TAM used to spend one hour a week working as a phone operator for his own company, taking up customer calls like any other operator. This was his way of keeping in touch with his customers. I always thought this was a brilliant idea.

Being a teacher is my way of keeping in touch with the students and the fellow instructors. If you want to build an education company, you must teach and learn on a daily basis.

As for what I teach, I specialize in iOS development, both with Objective C and Swift. Right now, I'm working on a future course of “Full Stack Swift”: creating both the backend and iOS clients for Apps with Swift. That's what I am looking forward to, as well as incorporating more functional programming concepts into our courses.

[BSD Mag]: Which concept do you see as the most difficult to understand by your students?

[FR]: Oddly enough, a pretty simple one: the delegate. This is a design pattern very common in Cocoa (the set of libraries used to develop for iOS and MacOS). Perhaps because it's not frequently used in other environments, it takes some time to sink in.

[BSD Mag]: And the concept that is in turn the most difficult to teach?

[FR]: I'd say the same. The best way to approach concepts that are hard to teach is by using real world analogies. Once you find the right one, it all goes smoothly. And if it's a nerdy analogy, so much the better. I tend to use a lot of Star Wars related examples: it works great and keeps the students entertained through the process. ;-)

[BSD Mag]: But you are not only teaching in KeepCoding?

[FR]: I have taught the Big Nerd Ranch's Advanced iOS Bootcamps in Europe, LATAM and US. I'm also collaborating with Udacity in their iOS Nanodegree program here in the US. This allowed me to learn other teaching methodologies. I believe this is a good thing.

[BSD Mag]: You are doing many things at the moment. How do you manage to be involved in so many companies and projects?

[FR]: With great difficulty and the invaluable assistance of Mr Steven Pressfield. ;-) Everything boils down to managing your time correctly and being able to save your "prime time" from interruptions.

For example, I always plan my day the night before. If you start your day by planning, you already lost the battle: emails are popping up, notifications, phone calls, requests for help by co-workers, etc. To survive, you immediately enter a "crisis mode" and once all the fires have been put out, no real work has been done.

My prime time is the morning, that's when I'm the most productive for creative work: programming, designing course materials or writing articles. During that time, I'm 100% unplugged and follow my to do list by the book.

During the afternoon, I plug all devices in, and deal with emails, Twitter, slack and whatever, but I know that I've done my work and now can devote some time for less important chores. Before the day ends, I plan the next day. When you're planning tomorrow's tasks, you have some distance from those future needs, distance enough to clearly see what's important and what's urgent. Always do what's important.

If you're struggling with too many things to do, by all means read "The War of Art" by Steven Pressfield. It's a brilliant book that I recommend to all my students.

[BSD Mag]: You have quoted Robert Frost on your LinkedIn profile: "I am not a teacher, but an awakener." What does it mean for you?

[FR]: There's a very interesting parable about the building of the pyramids:

An old man sees 3 men working on the pyramids in ancient Egypt. He asks the first one, "What are you doing?" The first man replies, "I'm laying bricks can't you see it?"

He then asks the second one: "I'm building a wall with bricks."

He then asks the last one: "I'm building a monument so magnificent that for millennia, men will look to it with awe!"

[BSD Mag]: Do you think that on-site training programs have an advantage over online courses?

[FR]: Yes, they do, but not for the students. An on-site training is vital for the instructor as it allows you to watch the students while they learn. This allows you to discover what are the bottlenecks, what parts are hard to grasp, what analogies work and which don't. An on-site training is the debugger for an online one. You must go through it before releasing a new online course.

We always do a few on-site sessions before moving ahead to an online version.

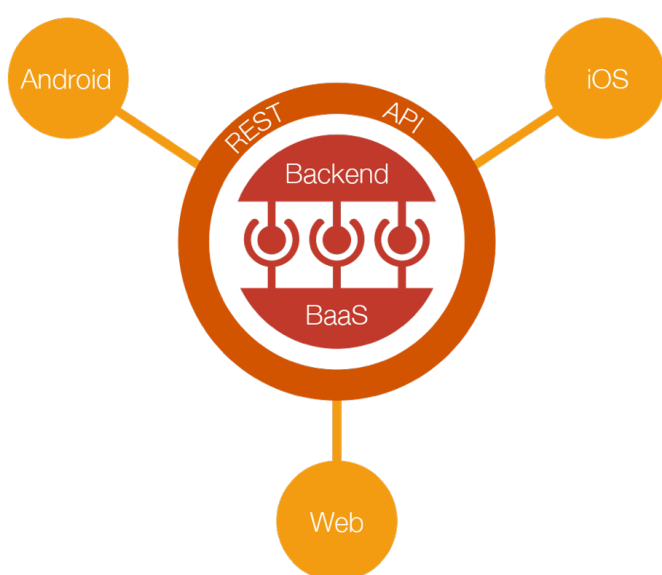
[BSD Mag]: What does Engineering Master Bootcamp look like? Do you take people to the forest in the middle of nowhere, tell them to code for a whole night and hunt during the days?

[FR]: I wish! :-) We actually work with what we call a blended model. We mix onsite activities where the students get to know each other as well as their instructors. It includes some fun stuff, but not hunting in the woods with a flint knife. I like the idea though, so we might consider it in the future! The rest of the classes are either online (with a schedule that must be followed and in real time with the instructor), or video lessons that the student can take at his own pace, but with deadlines.

It's a very intensive program and not for the faint of heart. However, the "survivors" have been very satisfied and the only metric I was willing to consider, job placement and employability, has been a resounding success.

[BSD Mag]: What is the most popular course? What do you think are upcoming trends, which skills will be the most desirable in the nearest future?

[FR]: Right now, our bestselling products are the two tracks in our Startup Engineering Bootcamps. Both last eight months are extremely practical and created for the real needs of the industry. Think of any successful digital product right now, say Facebook, Whatsapp, Evernote, Uber, AirBNB, Netflix, you name it. They all share a common architecture, with a backend (usually a series of micro services working together), a REST API and several clients, web and mobile, usually.



This is the basis for our bootcamps, the students learn how to build each component and how to integrate it within the whole system. They end up by creating a realistic market (think of a super local eBay) application MVP with all the components. This last part is done as a group, working with scrum. Some groups have even decided to apply their own business plan and create a startup.

The main difference between both tracks, mobile and web, is the emphasis on each technology.

I believe that both technologies, mobile and web, are safe bets for the foreseeable future. The success of one technology pulls the other: any successful mobile App needs a backend and vice versa.

These are great times for being a developer!

[BSD Mag]: What do you think about open source? Are you a fan of its communities, very devoted to DevOps?

[FR]: I'm a big fan of Open Source and I consider it one of the greatest cultural advancements of the past century. I believe the great challenge now is to port this same concept to other fields of human knowledge.

Some limited experiments have been carried out in the past and with great success. In 1959, Nils Bohlin, a Volvo engineer, "open sourced" the company's most important invention ever: the three point seat belt. This brought more visibility to Volvo than any marketing campaign could have achieved and has saved innumerable lives since then.

Hopefully, the XXI century will see the Open Source approach entering other areas, while preserving the right for intellectual property.

[BSD Mag]: What do you see as the biggest advantage of the open source approach?

[FR]: One of the best ways to become a better engineer, software or not, is to become part of two types of teams.

First, you should join on where you're the least knowledgeable member. Just by observing senior engineers solving problems and doing their work, you will gain insights and knowledge that no book will ever be able to teach you. It's like watching two Grand Masters play while you're learning Chess.

The other group you must join is one where you are the senior member. It will teach you to lead and manage.

Thanks to the Open Source movement, we software engineers have enormous ease of doing this and tapping into the wisdom of the gods in our field. This is invaluable and I always

recommend my students become involved in some open source project in their field of interest.

[BSD Mag]: What is the biggest challenge for your company at the moment? What is the biggest challenge for the industry?

[FR]: We started with very small groups of students, devoting an enormous amount of attention to their needs. Almost all of our growth has been by word of mouth from delighted students. As personally satisfying as it was, this model was not scalable.

We have improved on it and keep working on being able to provide a customized learning experience of the greatest quality to as many students as possible at a reasonable cost.

This is the great challenge of all the education industry. This is compounded here in the US with the enormous cost of traditional education, that can sometimes bankrupt students. As surprising as it may be to a European reader, many young Americans are fleeing the country because they cannot pay their student loans.

The education industry is on a tipping point and is ripe for a complete disruption. The change will start here in the US, where the problems are more poignant, but it will change the global landscape. My five year old son will probably never have a desktop computer or an internal combustion car. He will likely not have a college degree, nor need it.

[BSD Mag]: Do you think it is better to have very specialized employees or people who can combine knowledge from many different branches of sciences and different skill sets, like business, marketing, coding and multimedia design, for example?

[FR]: If we're talking about software development, the ability and willingness to learn is far more important than anything else. A very specialized employee will be useless to the company and to himself in four years if he doesn't reinvent himself periodically.

In a startup, a "specialized generalist", someone who knows a lot about a specific topic but also understands other aspects of the business, is invaluable.

If you're a mobile developer, but also have a sound knowledge of backend development, if your sprint is done, you can help the backend team to cleanup their backlog. This makes you more valuable to the company.

Having an understanding of the big picture is also invaluable for yourself, as it "increases the reach of your radar": you can detect changes in the different technology trends and adapt quicker before becoming outdated.

This is why, in our bootcamps, both the web and mobile ones, we don't limit ourselves to teaching programming "hard skills". We include design, agile and business literacy tracks.

[BSD Mag]: Do you have any advice for our readers?

[FR]: The name of our company, KeepCoding, was inspired by a Johnny Walker ad I once saw in Beirut many years ago. It was created after the Israeli air force had destroyed many bridges in the south of Lebanon. It displayed Johnny, walking as usual, over a broken bridge. It said: “Keep Walking”. That’s it. Nothing else.

I loved it and felt it expressed my own beliefs: no matter what, keep walking. So no matter what the difficulties might be, keep walking. And if you are a developer, and programming is what you were born to do, by all means, Keep Coding!



About Fernando:

Fernando Rodríguez has 20 years of experience as a developer and teacher. He is a co-founder of KeepCoding, a training company based in Madrid and Berkeley. He has trained 5000+ developers around the world, from Facebook to indie devs, both onsite and online. His online iOS course has been mentioned on Financial Times, VentureBeat and Information Week.

One of these years he will be able to devote full time to his real passion and talent: cooking!

Baroness Neville-Rolfe DBE CMG, in the recently released UK government document “Criminal Sanctions for Online Copyright Infringement” mandates a 10 year prison sentence for serious instances of copyright infringement. This intends to bring the penalties in line with those found guilty of copyright breaches in respect to physical goods. Will this amendment help to reduce piracy?

by Rob Somerville

At first glance, harmonizing the penalties for physical and digital copyright infringement would seem to be a logical step. Artists and business have every right to expect legal protection from those that would exploit their investment – time, emotional, mental, financial or otherwise – and make substantial profits off the back of their creativity. Defining where we draw the line between culture, fraud and theft, however, is a much more difficult concept. And this is where the whole issue of licensing and IP rights falls flat on its face. When does a creation become public domain? How much protection should we give to work X where this is based on cultural input from others? How original is an original work?

The derision that was served on Apple Inc. concerning their lawsuit against Samsung about rounded corners is a classic case in point. This totally mendacious lawsuit is a good example of the real undercurrents that lie beneath the surface of any technology industry. As riches are to be made, the vultures

in the form of lawyers, accountants, PR departments and vested interests gather together in an unholy coalition to annihilate any vestige of common sense, jurisprudence, natural law and legal precedent.

Having read the document signed by the Baroness, I personally don't think that her motives are totally misplaced. Organized crime is an evil that needs to be addressed, and anyone who doesn't understand the level of penetration that society has suffered from needs their head examined. Until the pond life that steals somebody's identity is banged up in the big house for 10 years, maybe then I will have the heart to support a change in legislation that allows somebody to be prosecuted under the legal framework of a strict liability offense (e.g. no mens rea defense) and potentially receive a sentence in some cases greater than those who have caused loss of life, disfigurement, or – shall we be cynical – have financially raped society by “legal” means?

It is proposed that protection will be in place to isolate individual offenders from such a draconian sentence. Your average geek in his / her basement sharing files via torrent can expect cease and desist notices, domain cancellation and, while not stated in the document, I suspect fines. To quote, the level of penalty should meet the scale of the crime. And therein lies the rub.

Counterfeiting is as old as prostitution. The only judicial difference between the two professions is the penalty if you are successful in your chosen vocation. The problem is that in the digital age, I can make a copy that is electronically indistinguishable from the original at byte level at the touch of a mouse. So the whole argument about software piracy, theft, etc. is a cultural misnomer that those who have a vested interest would prefer to propagate. Theft is based on the assumption that you deprive the legal owner of use of X by spiriting it away. Breaking IP law – worst case – should be considered fraud. Yet the penalties for counterfeiting currency or historical art are traditionally far more severe than those for the pimp.

In the final analysis, this proposed piece of legislation has the stench of vested interest about it. It is well known that there are counterfeit parts aplenty in the supply chain of critical industries – automotive, airline, IT and nuclear. These components, should they fail, could cause serious injury or death. If some plutocrat is ripped off by buying a fake painting, or the economy takes a hit by a craftsman who can recreate a viable currency note, this is fraud. An innocent person going about their

daily life dying in a car crash due to faulty parts, or a child suffering burns due to a counterfeit power supply or battery is a different matter entirely.

The level of fraud and corruption in the West has reached proportions that would cause our great-grandparents to turn in their grave. Little focus, or indeed financial support, is given to agencies that understand the true scale of what is going on in our globalized, technology driven society. As a teenager, I abundantly recorded music off the radio onto cassette tape. What concerns me more is the school colleague I knew at the time in my first Saturday job. A prolific thief, everything from expensive metal tapes to televisions were spirited away from the major high street store to be resold at a profit. Last time I heard, he was a detective with the local police force.

To quote “Penalty Fair? Study of criminal sanctions for copyright infringement available under the CDPA 1988”, the issue boils down to one simple question. Fundamentally, either online copyright offenses are capable of causing serious harm, or they are not. The difficulty arises right at the very top level of the legal tree – is this offense civil or criminal? Whereas in criminal law the guilty decision has to be made beyond all reasonable doubt, civil law is based on balance of probabilities. The inclusion of strict liability in this proposed legislation blurs an already muddy landscape.

I have stated many times we have too much law, too many hooks upon which the necks of the relatively innocent may be hung.

It has come to the point that we will all go to our graves law breakers. The key question is whether or not we would prefer to be governed by the letter rather than the spirit of the law. Looking at the corporate IT sector, the former seems to be the trend. That is what you get for subscribing to the ethic of knowing the price of everything, but the value of naught.

When it comes to judicial punishment, there is only one way to guarantee an offender does not repeat their offense ever again and that is to dispatch them – swiftly or otherwise – into the next life. Many countries have now repealed the death penalty, and offenses depending on their severity are met with everything from a verbal caution to a lengthy spell in prison. The punishment should fit the crime. One suspects that the zeal with which the Minister for Intellectual Property has approached the whole sentencing issue has more to do

with protecting corporate wealth than true justice.